Titre: Title:	Application de la synthèse H∞ structurée au co-design d'actionneurs et de lois de commande de satellites
Auteur: Author:	Pierre Daligault
Date:	2019
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Daligault, P. (2019). Application de la synthèse H∞ structurée au co-design d'actionneurs et de lois de commande de satellites [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/3947/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/3947/
Directeurs de recherche: Advisors:	David Saussié
Programme: Program:	Génie aérospatial

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Application de la synthèse $\mathcal{H}_\infty$ structurée au co-design d'actionneurs et de lois de commande de satellites

PIERRE DALIGAULT
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie aérospatial

Juillet 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Application de la synthèse $\mathcal{H}_\infty$ structurée au co-design d'actionneurs et de lois de commande de satellites

présenté par **Pierre DALIGAULT**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Richard GOURDEAU, président
David SAUSSIÉ, membre et directeur de recherche
Guchuan ZHU, membre

DÉDICACE

À toutes les personnes formidables que j'ai eu la chance de rencontrer à Montréal,
merci d'avoir rendu chaque instant inoubliable ...
vous me manquerez !

REMERCIEMENTS

Je tiens à remercier tout particulièrement mon directeur de recherche M. David Saussié pour son soutien et sa patience et qui aura su me guider tout au long de ma maîtrise recherche. Je remercie ensuite M. Richard Gourdeau et M. Guchuan Zhu pour avoir accepté de lire mon mémoire et de participer à ma soutenance en tant que membre du jury. Je leur suis reconnaissant à tous les trois pour les cours qu'ils ont dispensés durant lesquels ils ont su partagé leur passion et enthousiasme au sujet du Contrôle des Systèmes et pour avoir confirmé mon grand intérêt pour le domaine.

J'aimerais aussi remercier mes parents et frères et soeurs dont les encouragements et soutiens outre-Atlantique a été une source intarissable de réconfort. J'ai également une pensée particulière pour mes amis et collègues de laboratoire Gabriel Laparra et Vincent Dabin qui par leur présence ont ensoleillé mes journées d'étudiant-chercheur ainsi que tous mes amis qui ont toujours su me soutenir et sans qui Montréal ne serait pas la même. Il me semble important de remercier Pierre Lallet avec qui j'ai eu le plaisir d'écrire un article et qui aura su par sa passion faire avancer les travaux développés dans ce mémoire. Petite mention également à mes chers colocataires Martin Colon et Fanny Lafond qui savent me supporter au quotidien et dont les encouragements m'ont poussé tout au long de cette maîtrise.

Un grand merci à vous tous !

RÉSUMÉ

L'optimisation des performances du système d'orientation est un enjeu majeur dans la conception des satellites. La démarche standard est constituée de deux étapes. Dans un premier temps, la conception des pièces mécaniques de telle sorte qu'elles soient dimensionnées au mieux par rapport aux efforts auxquels elles seront confrontées. Ensuite, le système de commande est synthétisé afin d'assurer que le comportement du satellite respecte bien les contraintes du cahier des charges que cela soit en temps de réponse, en précision de pointage ou limitation de consommation en énergie par exemple. Les performances ainsi obtenues sont souvent satisfaisantes, mais de récentes études poussées par la mise au point de nouvelles synthèses de contrôleur comme la synthèse $\mathcal{H}_\infty$ structurée ont montré qu'une conception intégrée pourrait être envisagée : le co-design. Au lieu de séparer conception mécanique et synthèse de correcteur, il s'agit ici d'optimiser ces deux systèmes en parallèle dans le but d'obtenir des performances optimales.

L'objectif de ce projet de recherche est donc de développer une démarche visant à optimiser en parallèle la structure mécanique des actionneurs et la commande en orientation d'un satellite afin d'obtenir des performances supérieures.

Dans une première partie, on présente la modélisation d'un satellite en orbite autour de la Terre, en prenant soin de bien inclure la mécanique associée aux actionneurs embarqués ; ce modèle est ensuite linéarisé pour les besoins de l'étude. Une seconde partie est consacrée à la présentation de la synthèse $\mathcal{H}_\infty$ structurée et de la théorie sous-jacente afin d'introduire les fonctions MATLAB utilisées dans ce mémoire pour réaliser le co-design du système de commande en orientation du satellite.

Ensuite, une démarche est développée à partir de ces fonctions MATLAB afin de démontrer l'efficacité et l'intérêt de la méthode proposée. Le co-design est alors appliqué au cas de la commande en orientation d'un satellite en réduisant la consommation en puissance d'un système composé de roues de réaction. Les fonctions utilisées sont détaillées afin d'éclaircir le fonctionnement de `systune` et comment cette fonction est appliquée au co-design de systèmes. De plus, l'influence du nombre de paramètres réglables est étudiée afin de conclure sur l'intérêt du co-design par rapport à la conception classique.

Finalement, un autre exemple d'application est présenté dans le cas d'un système d'orientation d'un nanosatellite. Il est montré que la méthodologie mise en place permet d'obtenir des dimensions des actionneurs et des gains du correcteur optimisés directement à partir des contraintes du cahier des charges dans un souci de réduction de masse et d'espace occupé.

ABSTRACT

Optimizing the performance of the orientation system is a major challenge in satellite design. The standard approach consists of two steps. First, the design of the mechanical parts so that they are adequately dimensioned in relation to the forces with which they will be confronted. Then, the control system is synthesized to ensure that the satellite's behaviour complies with the requirements of the specifications, whether in terms of response time, pointing accuracy or energy consumption limitation, for example. The performances thus obtained are often satisfactory, but recent studies pushed by the development of new controller syntheses such as the structured $\mathcal{H}_\infty$ synthesis have shown that an integrated design could be considered, namely co-design. Instead of separating mechanical design and controller synthesis, the aim here is to optimize these two systems in parallel in order to obtain optimal performance.

The objective of this research project is therefore to develop an approach aimed at optimizing in parallel the mechanical structure of the actuators and the orientation control of a satellite in order to obtain superior performance.

In the first part, we present the modeling of a satellite in orbit around the Earth, taking care to include the mechanics associated with on-board actuators; this model is then linearized for the purposes of the study. A second part is devoted to the presentation of the structured $\mathcal{H}_\infty$ synthesis and the underlying theory in order to introduce the MATLAB functions used in this thesis to realize the co-design of the satellite orientation control system.

Then, an approach is developed based on these MATLAB functions to demonstrate the effectiveness and interest of the proposed method. Co-design is then applied to the case of satellite steering control by reducing the power consumption of a system composed of reaction wheels. The functions used are detailed to clarify how `systune` works and how this function is applied to system co-design. In addition, the influence of the number of adjustable parameters is studied in order to conclude on the interest of co-design compared to conventional design.

Finally, another example of application is presented in the case of a nanosatellite orientation system. It is shown that the methodology implemented makes it possible to obtain optimized actuator dimensions and controller gains directly from the constraints of the specifications in order to reduce mass and occupied space.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ADCS	Attitude and Determination Control System
DCM	Direct Cosine Matrix
ECI	Earth Center Inertial
IGRF	International Geomagnetic Reference Field
IMU	Inertial Measurement Unit
ISS	International Space Station
LFT	Linear Fraction Transformation
LMI	Linear Matrix Inequality
LQR	Linear Quadratic Regulator
MIMO	Multiple Input Multiple Output
PD	Proportionnel-Dérivé
PID	Proportionnel-Intégral-Dérivé
SCAO	Système de Commande d'Attitude et d'Orbite
SISO	Single Input Single Output

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contexte

L'Espace n'a jamais tant été autant à la portée de l'Homme. Les récents progrès technologiques et initiatives ont permis aux entreprises privées et aux universités de construire et de placer en orbite leurs propres satellites alors qu'il y a quelques décennies cela était encore réservé aux agences spatiales. L'émergence des nanosatellites tels que les cubesats et les récents succès de Space X à développer des lanceurs réutilisables en sont les meilleurs exemples ; ceux-ci ont permis de baisser drastiquement les coûts autrefois nécessaires au développement des missions spatiales. Ainsi, si l'optimisation des systèmes embarqués d'un satellite a toujours été un enjeu crucial depuis le début des missions spatiales en 1957, cela est d'autant plus le cas aujourd'hui où les systèmes toujours plus performants sont aussi confrontés à des problématiques de restriction des coûts.

Un des sous-systèmes les plus importants et source de nombreuses études est le système de commande d'attitude et d'orbite (SCAO), plus communément appelé en anglais AOCS pour *Attitude and Orbit Control System*. Celui-ci est en effet responsable d'assurer la stabilité du satellite et ses performances que ce soit en précision de pointage ou ralentissement de sa vitesse de rotation. Il serait impossible de concevoir un satellite sans SCAO car il ne pourrait pas fonctionner une fois en orbite. Afin de permettre au satellite d'accomplir ses missions, le SCAO va donc devoir compenser les perturbations externes qui s'appliquent sur le satellite tout au long de son orbite et corriger en permanence son orientation en utilisant les différents actionneurs implantés dans le satellite.

De nombreux types d'actionneurs ont été développés et perfectionnés au fur et à mesure de la conquête spatiale, chacun présentant ses propres caractéristiques et avantages. Les plus répandus demeurent néanmoins les roues de réaction et les magnéto-coupleurs pour le contrôle en orientation ainsi que les systèmes de propulsion qui permettent également de modifier l'orbite du satellite. Toutefois, c'est grâce aux algorithmes de commande que les ingénieurs et les chercheurs sont parvenus à améliorer les performances par des lois de commande toujours plus complexes. C'est au cours du XX^{e} siècle et des guerres mondiales que les premiers systèmes aérospatiaux sont apparus, et avec eux les premiers correcteurs. Connus sous le nom de correcteurs PID (Proportionnel-Intégral-Dérivé), ils sont toujours largement utilisés de nos jours, et ce dans tous les domaines de l'automatique, que ce soit en robotique ou en

commande des systèmes aérospatiaux (incluant les satellites [5]). Toutefois, les systèmes à commander étant toujours plus complexes et les contraintes de conception toujours plus restrictives de nouvelles alternatives pour la synthèse des contrôleurs ont vu le jour. C'est le cas de la synthèse $\mathcal{H}_\infty$ qui est apparue au courant des années 1980 qui a permis de poser les bases de ce qui est à présent appelé la commande robuste [6, 7]. Ce type de synthèse a permis de répondre à l'émergence de nouveaux défis tels que la sensibilité aux incertitudes [8] ou encore la commande dans le domaine fréquentiel pour atténuer les vibrations dues aux panneaux solaires [9]. Plus récemment encore, P. Gahinet et P. Apkarian ont entrepris de regrouper les avantages de ces deux méthodes de commande en une seule en développant la synthèse $\mathcal{H}_\infty$ structurée [10]. Celle-ci propose une approche innovante qui tente d'obtenir des performances de robustesse avec une architecture de correcteur fixée *a priori*, e.g., un correcteur PID bien plus simple à implémenter pour un SCAO qu'un correcteur d'ordre élevé.
Grâce à ces correcteurs, il est alors possible d'imposer des critères de performance de différentes natures au satellite tels que la réponse temporelle du système (en termes de temps de réponse, de dépassement de consigne, etc.) ou encore imposer un gabarit fréquentiel pour par exemple atténuer l'amplitude de vibrations internes au système.

1.2 Objectifs de la recherche

Ce mémoire s'inscrit à l'origine dans le projet de réalisation d'un nanosatellite de type Cubesat au sein de la société technologique PolyOrbite de Polytechnique Montréal. Le but était alors de réaliser le système de commande en orientation constitué de roues de réaction et de magnéto-coupleurs. Bien que le projet ait finalement été modifié de telle sorte que ce système soit simplifié en un aimant permanent faute de budget, un chapitre sera consacré à l'étude réalisée pour ce cubesat. Dans le but de minimiser au possible les dépenses associées, il avait été décidé de construire les roues et les bobines des magnéto-coupleurs au sein de l'université plutôt que d'acheter celles-ci. C'est ainsi qu'est venue l'idée de chercher à optimiser les actionneurs selon les performances que nous cherchions à obtenir afin de ne pas avoir un système surdimensionné, trop volumineux ou trop lourd.

L'objectif principal de ce mémoire est donc de développer une méthode permettant d'optimiser les actionneurs en parallèle de la synthèse du correcteur en orientation d'un satellite. En pratique on parlera de co-design des actionneurs et du contrôleur.

Comme cela sera évoqué dans la suite, peu de travaux abordent ce sujet et il s'avère complexe de trouver une méthode claire permettant de réaliser le co-design d'un système. Un des objectifs de ce mémoire est donc de développer une démarche simple et accessible permettant d'effectuer la conception en parallèle du correcteur et de la structure du système à l'aide de

MATLAB. Une seconde partie sera ensuite dédiée à l'application de cette démarche à un projet de conception de nanosatellite pour confirmer l'efficacité de la méthode proposée.

1.3 Plan du mémoire

Le mémoire se présente de la manière suivante. Le chapitre 2 est dédié à la revue de littérature afin de présenter différents travaux ayant permis à ce projet de se réaliser, puis le chapitre 3 est dédié à la présentation de la synthèse $\mathcal{H}_\infty$ puis de l'avancée que permet la synthèse $\mathcal{H}_\infty$ structurée dans le domaine du co-design. Le chapitre 4 est ensuite consacré à la modélisation des satellites et de leur environnement. Le chapitre 5 présente les résultats de l'article écrit dans le cadre de ce mémoire et présenté à la conférence AIAA Space 2018 tandis que le chapitre 6 nous permet d'appliquer la méthode de co-design développée dans ce mémoire à un projet de nanosatellite. Finalement, dans le chapitre 7, nous concluons sur les résultats obtenus.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de littérature afin d'introduire les différents sujets qui seront abordés dans le mémoire. L'objectif de ce mémoire étant d'optimiser le système de commande en orientation d'un satellite, il est nécessaire de présenter les avancées dans les deux principaux sujets concernant le co-design, à savoir la modélisation des satellites ainsi que leurs actionneurs, et la synthèse des algorithmes qui les commandent. La dernière partie de cette revue de littérature est dédiée aux travaux précédents qui ont cherché à appliquer le co-design aux satellites.

2.1 Satellites

La modélisation des satellites a été largement traitée par de nombreuses études au cours des dernières décennies. Il s'agit en effet d'une étape cruciale pour la synthèse de correcteurs, une trop grande erreur de modélisation menant très souvent à des performances éloignées de celles souhaitées, voire à un système instable. Nous allons donc détailler dans cette section comment la dynamique des satellites est modélisée dans la littérature scientifique, puis les principaux systèmes de SCAO (actionneurs et capteurs) seront présentés avant de voir comment ceux-ci peuvent être paramétrés.

2.1.1 Modélisation des satellites

Les agences spatiales telles que la NASA [1] ont joué un rôle très important dans le développement des technologies spatiales ; c'est sous leur impulsion que de nombreux travaux scientifiques sont apparus dès la seconde moitié du XX^{e} siècle, modélisant de façon toujours plus précise le comportement dynamique des satellites. On peut à cet égard nommer [11] ou encore [12]. De nombreux ouvrages tels que [13] détaillent les différentes méthodes de modélisation d'attitude ainsi que les différentes perturbations environnementales à considérer pour obtenir une modélisation le plus proche possible de la réalité. D'autres travaux se sont concentrés sur les cas plus complexes comme les structures spatiales avec panneaux flexibles, qui induisent des modes de vibrations [14], ou avec des voiles solaires [15].

Avec l'émergence des nanosatellites, l'intérêt pour le sujet de la commande des satellites ne cesse d'augmenter ce qui a permis d'accroître la précision des modèles mécaniques et dynamiques et de développer de nouvelles techniques de stabilisation, par exemple l'utilisation des perturbations environnementales pour stabiliser les satellites [16]. On peut également

remarquer que de nombreuses études universitaires se sont concentrées sur les nanosatellites de type Cubesat (Fig. 2.1) afin d'obtenir des systèmes plus performants, mais également moins dispendieux dans le cadre de projets étudiants [17].

Figure 2.1 Nanosatellite de type Cubesat 1U (source : nasa.gov)

2.1.2 Mécanique des actionneurs et détermination d'attitude

Les actionneurs sont des systèmes essentiels pour l'orientation des satellites. Ce sont eux qui permettent le mouvement en transformant l'énergie électrique ou chimique par le principe d'action-réaction de la troisième loi de Newton. Ceux-ci sont commandés par des lois de commande qui permettent d'assurer la stabilité et les performances du satellite tout au long de sa mission.

Parmi les actionneurs les plus courants pour commander l'attitude d'un satellite se trouvent les roues de réaction et les magnéto-coupleurs, représentés respectivement dans les figures 2.2 et 2.3. Ceux-ci consomment peu d'énergie et ont l'avantage de pouvoir être utilisés continuellement tant que les panneaux solaires collectent de l'énergie, en comparaison avec les propulseurs (Fig. 2.4) qui ont une utilisation limitée au volume de propergol contenu dans un réservoir, souvent massif.

La comparaison entre différents actionneurs a été par exemple traitée dans [3] et [1] qui ont cherché à comparer quels types d'actionneurs utiliser en fonction des objectifs des missions du satellite. Une comparaison des actionneurs est fournie dans le tableau 2.1 ; on peut y voir les différentes précisions de pointage accessibles et les facteurs limitant leur utilisation. La

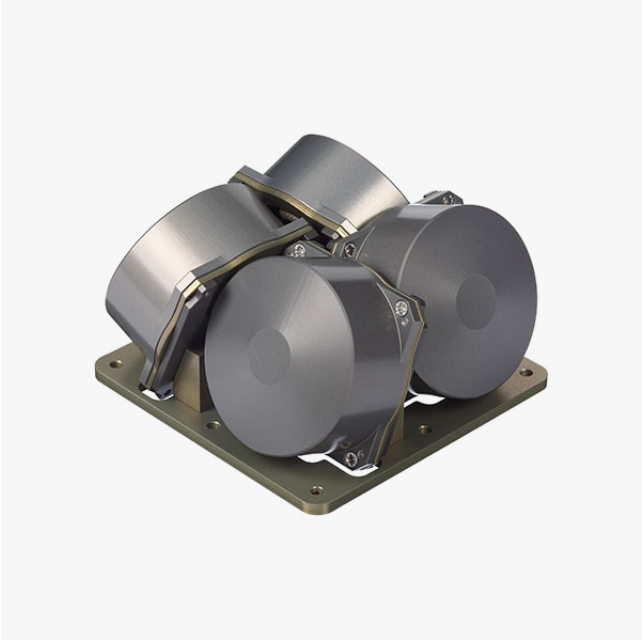

Figure 2.2 Assemblage de quatre roues de réaction (source : n-avionics.com)

Figure 2.3 Disposition de trois magnéto-coupleurs (source : n-avionics.com)

Figure 2.4 Propulseur à Hydrazine (source : ariane.group)

précision de pointage peut varier de façon importante selon la précision des capteurs utilisés, ce qui explique la grande plage de variation pour les roues de réaction notamment.

Tableau 2.1 Comparaison des actionneurs (tiré de [1])

Actionneurs	Précision de pointage	Durée de vie
Roues de réactions	$\pm 0.0001°$ à $\pm 5°$	Durée de vie des capteurs ou des roulements
Magnéto-coupleurs	$\pm 1°$ à $\pm 5°$	Durée de vie des capteurs
Propulseurs	$\pm 0.1°$ à $\pm 5°$	Quantité de propergol (carburant)

Il existe une grande variété de capteurs permettant de déterminer l'attitude d'un satellite. Un des choix les plus communs est la centrale inertielle ou IMU (pour *Inertial Measurement Unit*) qui permet de mesurer la vitesse angulaire (grâce à trois gyroscopes) et l'accélération (grâce à trois accéléromètres). En intégrant ces mesures, il est alors possible d'estimer l'attitude du satellite à partir de sa position initiale. Toutefois il est souvent nécessaire d'utiliser d'autres capteurs en parallèle afin de déterminer précisément l'orientation du satellite [18]. Les principaux capteurs utilisés en complément des IMU sont donnés dans le tableau 2.2 ; le choix se fait souvent en comparant les requis du cahier des charges et le budget alloué.

Tableau 2.2 Capteurs SCAO (tiré de [1])

Capteurs	Précision
Capteur solaire	$0.005°$ à $3°$
Capteur stellaire	$0.0003°$ à $0.001°$
Capteur d'horizon	$0.05°$ à $1°$
Magnétomètre	$0.5°$ à $3°$

Il ressort ainsi que les roues de réaction permettent d'obtenir une précision d'orientation bien supérieure à celles des autres actionneurs, ce qui en fait un choix très courant pour

les systèmes de pointage ou optique [19]. Toutefois, celles-ci sont particulièrement lourdes surtout dans le cas des nanosatellites dont elles constituent une grande partie de la masse totale.

De nombreux travaux tels que ceux de [20] et [21] ont proposé plusieurs façons de modéliser le fonctionnement des roues afin de les rendre toujours plus fidèles au comportement réel et de pouvoir optimiser leur conception au plus proche des besoins du cahier des charges. D'autres études, quant à elles, se sont concentrées sur les lois de commande de ces roues, en particulier [22, 23] qui présentent une méthode pratique et précise pour le contrôle en courant et vitesse de rotation. Notons également que [24] a abordé la problématique de la dynamique des roues en incluant les frottements mécaniques afin de modéliser au plus près leur fonctionnement réel.

Bien que moins précis les magnéto-coupleurs sont néanmoins moins énergivores et sont souvent requis lorsque les roues de réaction arrivent à leur niveau de «saturation». Leur fonctionnement a été traité dans plusieurs travaux, et ceux-ci ont été utilisés à bord des satellites dès le début des missions spatiales. Ceci est par exemple abordé dans [25] et [26] qui détaillent les dynamiques associées à la commande complète du satellite à partir de seulement de trois magnéto-coupleurs.

2.1.3 Paramétrisation des actionneurs

Dans le but d'optimiser les performances des actionneurs, certains chercheurs se sont quant à eux concentrés sur la paramétrisation de ceux-ci et ont lié leurs caractéristiques géométriques aux performances des satellites.

La disposition des roues à l'intérieur du satellite est par exemple sujette à de nombreuses études, en changeant leur orientation voire en ayant plus de trois roues, il est possible de conserver un contrôle adéquat malgré la panne de l'une d'entre elles [27] et de réduire leur consommation en énergie [28]. De même, l'optimisation de la géométrie des roues permet d'en réduire la masse et de l'adapter plus précisément aux contraintes de conception [29].

Cela est également vrai pour la conception des magnéto-coupleurs : [30] a cherché à optimiser les performances des bobines en modifiant leur géométrie. De plus, [17] a appliqué cette méthode en reliant les nombres de tours de fil et dimensions de bobines aux capacités magnétiques des magnéto-coupleurs afin de les dimensionner pour une mission de Cubesat étudiant, ouvrant la voie à des conceptions de bobines plus proches des besoins réels.

2.2 Synthèse de correcteurs

2.2.1 Commande standard

La synthèse de correcteurs a toujours été un enjeu majeur depuis l'avènement de l'automatique qui s'applique à de nombreux domaines, dont celui de l'aérospatial. Comme avancé par [5], il existe de très nombreuses méthodes de synthèses de correcteurs, chacune présentant des avantages et des inconvénients qui leur sont propres.

Le plus connu des correcteurs est le correcteur PID (pour Proportionnel Intégral Dérivé) : celui-ci est massivement répandu du fait de sa simplicité et de la bonne compréhension de son fonctionnement. Dans sa forme basique, il s'agit en effet de régler trois gains K_p, K_i et K_d portant respectivement sur l'écart entre la consigne et la mesure de la sortie à commander (l'erreur), son intégrale et sa dérivée, jusqu'à ce que la réponse temporelle satisfasse les critères de performance imposés [31]. Ce correcteur demeure satisfaisant dans le cas de systèmes bien modélisés et dont les perturbations sont bien connues, mais se révèle moins performant lorsque l'on cherche à optimiser la consommation en énergie par exemple, ou à garantir une certaine robustesse.

Une approche moderne qui s'est grandement développée avec l'essor des missions spatiales est la commande optimale ou LQR (pour *Linear Quadratic Regulator*) qui repose sur le principe de la minimisation de fonctions reliées aux performances et aux niveaux de commande utilisés. Bien que moins intuitive à mettre en œuvre, elle permet une commande plus fine de la dynamique du système et des travaux toujours plus élaborés ont permis de perfectionner ce type de correcteur pour améliorer la stabilité selon les phases de fonctionnement du satellite [32].

2.2.2 Synthèse robuste

La synthèse robuste a été développée dans les années 1980 par Zames et Doyle [6,33] qui cherchaient alors à améliorer la robustesse des systèmes, c'est-à-dire assurer la stabilité malgré des perturbations extérieures ou encore des incertitudes dans la modélisation. Cette synthèse, également appelée synthèse $\mathcal{H}_\infty$, a permis de grandes avancées dans la commande des satellites, et demeure à l'heure actuelle celle qui est la plus souvent utilisée. Des travaux tels que ceux de [34] ou [7] ont largement contribué à son développement en simplifiant la résolution numérique des problèmes de synthèse $\mathcal{H}_\infty$, par exemple en les transposant sous la forme LMI *Linear Matrix Inequality*.

Un bon exemple d'utilisation cette synthèse peut être trouvé dans l'article de [35], où il est

montré qu'il est possible d'imposer des contraintes dans le domaine fréquentiel au système en boucle fermée afin d'atténuer les vibrations mécaniques au sein du satellite.

La synthèse $\mathcal{H}_\infty$ permet également de réaliser des optimisations multi-objectifs, c'est-à-dire que le système commandé est alors capable de respecter plusieurs contraintes préalablement fixées. Par exemple, cela peut être un fonctionnement dans différentes conditions d'opérations [36] ou encore minimiser des pics de consommation tout en garantissant les performances [37]. On comprend alors tout l'intérêt de ce type de synthèse dont les applications sont très vastes et permettent de pousser toujours plus loin les performances des systèmes aérospatiaux. Toutefois, cette synthèse présente pour inconvénient majeur que les correcteurs ainsi obtenus sont d'ordre très élevé ce qui rend nécessaire leur réduction une fois synthétisés. De plus, ceux-ci n'ont pas d'architecture particulière rendant plus complexe leur implantation dans les systèmes embarqués.

2.2.3 Synthèse $\mathcal{H}_\infty$ structurée

L'apparition récente de la synthèse $\mathcal{H}_\infty$ structurée a répondu aux inconvénients de la synthèse $\mathcal{H}_\infty$ classique. En effet dans les travaux réalisés par J. Burke [38], et plus tard ceux de P. Gahinet et P. Apkarian [39], les chercheurs sont parvenus à synthétiser des correcteurs robustes dont la structure avait été fixée *a priori*. On cherche alors à synthétiser un correcteur ayant une architecture simple, par exemple de type PID, dont les performances sont comparables à celles d'un système avec un correcteur robuste d'ordre élevé. Cette avancée majeure a permis une évolution dans la conception des systèmes de commande aérospatiaux. Des travaux tels que ceux de [40] ont d'ores et déjà prouvé l'efficacité de la synthèse en l'appliquant sur un satellite avec des panneaux flexibles en tenant compte des incertitudes sur les paramètres de modélisation. Il en a résulté un système en boucle fermée robuste sur tout l'intervalle d'incertitudes dont les modes flexibles étaient atténués.

Il est important de noter la contribution de [10] qui présente la démarche nécessaire afin de réaliser la synthèse $\mathcal{H}_\infty$ structurée d'un correcteur en utilisant les fonctions `hinfstruct` et `systune` de la *Robust Control Box* de MATLAB . En l'appliquant au cas d'un avion, cet article présente le premier véritable mode d'emploi pour obtenir relativement aisément un correcteur par ce type de synthèse. L'implantation de ces fonctions dans MATLAB peut être considérée comme le point de départ de la "démocratisation" de la synthèse $\mathcal{H}_\infty$ structurée.

Les avantages offerts par cette synthèse en font la méthode de prédilection pour la conception des systèmes d'orientation de satellites. On peut à cet égard mentionner la sonde spatiale Rosetta (construite par l'Agence Spatiale Européenne (ASE) dont le correcteur a été conçu grâce à la synthèse $\mathcal{H}_\infty$ structurée. [41] détaillent par ailleurs comment les chercheurs de l'ASE

ont pu appliquer cette synthèse afin de compenser les incertitudes relatives à la position du centre de masse ou encore à l'inertie totale de Rosetta.

Il apparaît ici très clair que la synthèse $\mathcal{H}_\infty$ structurée est avantageusement utilisée dans le but de garantir la robustesse du système malgré des incertitudes de modélisation. Cependant, quelques études récentes ont pu prouver qu'il était également possible d'étendre cette application à l'optimisation de paramètres.

2.3 Co-design

Les systèmes SCAO sont parmi les plus lourds et énergivores de tous les systèmes embarqués dans un satellite ; leur optimisation a donc toujours été un défi majeur et de nombreux travaux y ont été consacrés. La démarche classique pour la conception de tels systèmes est composée de deux étapes : dans un premier temps, les actionneurs sont dimensionnés pour répondre aux contraintes imposées puis la loi de commande est synthétisée afin que les performances temporelles ou fréquentielles soient celles souhaitées. C'est ainsi que l'idée de co-design est apparue dans le but de proposer une nouvelle étape dans les phases préliminaires de conception des satellites. Le but est ainsi d'optimiser en même temps la loi de commande et les actionneurs mécaniques afin d'obtenir des performances encore supérieures. Le co-design est un sujet encore assez peu abordé par des travaux de recherches. En effet, ce n'est que récemment que des avancées ont permis de simplifier les démarches et les temps de calcul nécessaires à cette discipline. Comme nous le verrons, son application est encore réduite aux domaines de l'aérospatial et de l'aéronautique, mais il est à prévoir que le co-design soit utilisé à des buts plus industriels dans les années à venir.

2.3.1 Optimisation de satellites

La conception des satellites était initialement limitée à des itérations de conceptions mécaniques puis de correcteurs, toujours de façons séparées. Certaines études telles que [42] ou encore [43] ont chercher à trouver une disposition des roues de réaction optimale pour limiter la consommation en énergie. Il est alors apparu que le problème d'optimisation s'avère de plus en plus complexe lorsque de nombreux paramètres sont variants, les temps de calcul étant alors souvent longs et le nombre de combinaisons de valeurs possibles trop important pour être traité.

C'est dans les années 1980 que Hale a posé les bases du co-design telles que connu aujourd'hui. Celui-ci a utilisé une fonction de coût à minimiser regroupant des paramètres géométriques et des contraintes de commande optimale permettant pour la première fois d'optimiser le

système mécanique et le système de commande [44]. Plus tard, [45] a également prouvé que les performances d'un satellite conçu grâce au co-design étaient supérieures à celles qui auraient été obtenues par une méthode classique.

2.3.2 Représentation LFT et co-design $\mathcal{H}_\infty$

Comme nous avons pu le voir dans la section précédente, la synthèse de correcteurs $\mathcal{H}_\infty$ a en partie été développée afin de compenser les éventuelles incertitudes de la modélisation des systèmes. La paramétrisation LFT (*Linear Fractional Transformation*) a permis de grandement simplifier la prise en compte de ces incertitudes et donc la robustesse des lois de commande des satellites [8,46]. Le principe est ici de créer une matrice $\mathbf{\Delta}$ constituée de toutes les incertitudes et de l'extraire du système nominal $\mathbf{P}(s)$ comme indiqué dans la figure 2.5. On souhaite alors synthétiser un correcteur qui assure minimalement la stabilité du système en dépit des incertitudes.

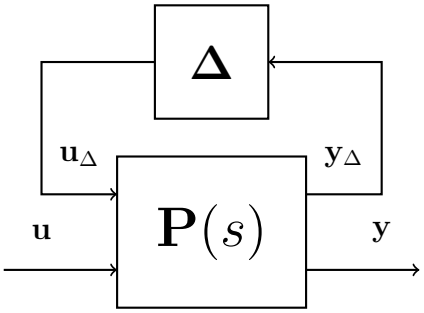

Figure 2.5 Représentation LFT

Hecker a pour sa part développé une méthodologie pour implémenter cette approche dans MATLAB et la $LFR-Toolbox$ [47]. Cette approche simplifiée est donc utilisée dans de nombreux travaux tels que ceux de [9], dans lequel un algorithme a permis de synthétiser un correcteur robuste et d'optimiser en parallèle le délai de transmission de commande d'un satellite.

2.3.3 Co-design $\mathcal{H}_\infty$ Structuré

Les algorithmes de synthèse $\mathcal{H}_\infty$ structurée ont permis de grands progrès dans le domaine du co-design également. Les travaux de [10] ont montré qu'il était possible de définir une architecture *a priori* afin d'optimiser les paramètres avioniques pour choisir les meilleurs capteurs et actionneurs possibles. De même, H.Lhachemi a appliqué cette même méthode pour déterminer la position optimale d'un capteur sur le fuselage d'un avion tout en garantissant une

loi de commande robuste sur toute l'enveloppe de vol [48].

Cependant, un meilleur exemple de l'avantage que représente la synthèse $\mathcal{H}_\infty$ structurée pour optimiser des structures aérospatiales est l'étude réalisée par J. Perez et D. Alazard [49]. Dans celle-ci, les chercheurs ont comparé plusieurs types de correcteurs : pour chacun d'entre eux, les mêmes contraintes de performances ont été imposées puis la masse maximale admissible pour le système embarqué a été calculée. Ils ont ainsi été capables d'optimiser la structure du correcteur pour augmenter la masse des systèmes embarqués tout en réduisant l'impact des perturbations sur le satellite.

On comprend donc ici que le co-design en est à ses balbutiements. Du fait de la complexité qui y était associée, peu d'études y ont été consacrées. Toutefois, la conception de la synthèse $\mathcal{H}_\infty$ structurée permet d'entrevoir un rapide retour en force du co-design dans de nombreux domaines de l'automatique. Ce mémoire vise à participer à la popularisation de cette méthode fortement prometteuse.

CHAPITRE 3 SYNTHÈSE $\mathcal{H}_\infty$ STRUCTURÉE

Ce chapitre vise à présenter la théorie derrière la synthèse $\mathcal{H}_\infty$ structurée ainsi que les algorithmes qui la mettent en œuvre. La première section est dédiée à la définition de la synthèse $\mathcal{H}_\infty$ standard afin de pouvoir introduire dans un second temps la synthèse $\mathcal{H}_\infty$ structurée, qui a pour avantage de permettre de fixer *a priori* l'architecture du correcteur. Enfin ,la dernière section présente l'algorithme MATLAB de `systune` que nous utiliserons dans la suite du mémoire pour réaliser le co-design du correcteur.

3.1 Présentation de la synthèse $\mathcal{H}_\infty$

La synthèse $\mathcal{H}_\infty$ a été développée dans les années 1980 par Zames et Doyle [6,7] afin d'assurer la stabilité et les performances des systèmes, malgré la présence d'incertitudes dans leurs modélisations ou de perturbations externes. Nous allons donc présenter dans cette section comment sont obtenus les correcteurs par cette synthèse et la théorie sous-jacente.

3.1.1 Formulation du problème $\mathcal{H}_\infty$

Avant de réaliser la synthèse du correcteur $\mathcal{H}_\infty$, il est souvent nécessaire de représenter le système sous sa « forme standard ». Cette formalisation permet de mettre en avant les performances que l'on souhaite imposer ainsi que les entrées et les sorties commandées.

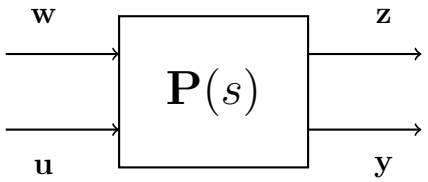

Figure 3.1 Forme standard

Comme nous pouvons le voir dans la figure 3.1, il est donc nécessaire de séparer les entrées du système $\mathbf{P}(s)$ en deux catégories :

- le vecteur $\mathbf{u}$ de commandes du système ;
- le vecteur $\mathbf{w}$ des entrées exogènes sur lesquelles des contraintes seront imposées lors de la synthèse du correcteur. Ces entrées peuvent être des perturbations ou bien des signaux de référence.

De même, les sorties sont divisées selon deux catégories :

— le vecteur $\mathbf{y}$ des sorties mesurées ;
— le vecteur $\mathbf{z}$ des sorties régulées que nous souhaitons commander ou optimiser.

Une fois écrite sous sa forme standard, la fonction de transfert du système augmenté $\mathbf{P}(s)$ peut être détaillée sous la forme ci-dessous. On obtient alors quatre matrices de fonctions de transfert liant $\mathbf{u}$ et $\mathbf{w}$ à $\mathbf{y}$ et $\mathbf{z}$.

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{y} \end{bmatrix} = \mathbf{P}(s) \begin{bmatrix} \mathbf{w} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{11}(s) & \mathbf{P}_{12}(s) \\ \mathbf{P}_{21}(s) & \mathbf{P}_{22}(s) \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{u} \end{bmatrix} \tag{3.1}$$

Avec cette représentation, l'implantation d'un correcteur $\mathbf{K}(s)$ entre les sorties mesurées $\mathbf{y}$ et les commandes du système $\mathbf{u}$ se représente alors comme montré sur la figure 3.2.

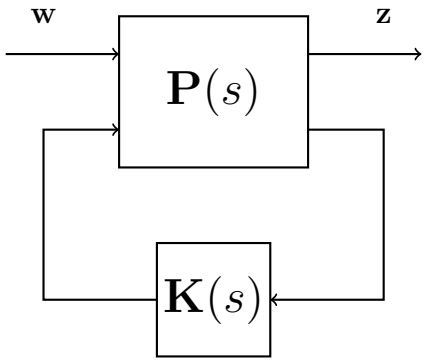

Figure 3.2 Problème $\mathcal{H}_\infty$

Une fois le système présenté ainsi, il est possible de trouver la fonction de transfert de $\mathbf{w}$ à $\mathbf{z}$ grâce à l'expression d'une LFT (*Linear Fractional Transformation*) :

$$\mathbf{T}_{\mathbf{zw}}(s) = \mathcal{F}_l(\mathbf{P}(s), \mathbf{K}(s)) = \mathbf{P}_{11}(s) + \mathbf{P}_{12}(s)\mathbf{K}(s)(\mathbf{I} - \mathbf{P}_{22}(s)\mathbf{K}(s))^{-1}\mathbf{P}_{21}(s) \tag{3.2}$$

3.1.2 Synthèse $\mathcal{H}_\infty$

La synthèse $\mathcal{H}_\infty$ a pour objectif de trouver un correcteur stabilisant $\mathbf{K}$ qui minimise la norme $\mathcal{H}_\infty$ de la matrice de transferts $\mathbf{T}_{\mathbf{zw}}(s)$.

Pour une matrice de transfert $\mathbf{T}(s)$, cette norme est notée $\|T\|_\infty$ et se calcule de la façon suivante :

$$\|\mathbf{T}\|_\infty = \max_{\omega \in \mathbb{R}} \bar{\sigma}(\mathbf{T}(j\omega)) \tag{3.3}$$

où $\bar{\sigma}(\mathbf{M})$ correspond à la valeur singulière maximale d'une matrice complexe $\mathbf{M}$.

Le but de la synthèse $\mathcal{H}_\infty$ est donc de chercher à minimiser la norme $\mathcal{H}_\infty$ de la fonction de transfert entre $\mathbf{w}$ et $\mathbf{z}$ afin de par exemple limiter l'impact de perturbations sur les performances du système tout en le stabilisant.

On peut résumer cela sous une approche formelle :

$$\begin{aligned} &\min \|\mathbf{T_{zw}}(\mathbf{P}(s), \mathbf{K}(\mathbf{s}))\|_\infty \\ &\text{s.c. } \mathbf{K}(\mathrm{s}) \text{ stabilise } \mathbf{P}(\mathrm{s}) \end{aligned} \tag{3.4}$$

On distingue cependant deux types de synthèse $\mathcal{H}_\infty$. La première est dite *optimale*, car on cherche le correcteur $\mathbf{K}(s)$ qui permet d'atteindre la valeur minimale de $\|\mathbf{T_{zw}}\|_\infty$, soit γ_{opt}. La seconde est quant à elle appelée *sous-optimale* car on cherche un correcteur $\mathbf{K}(s)$ qui amène la norme $\|\mathbf{T_{zw}}\|_\infty$ en dessous d'un certain seuil $\gamma > \gamma_{\text{opt}}$. En pratique, on préfère souvent avoir recours à celle-ci.

3.1.3 Inconvénients de la synthèse $\mathcal{H}_\infty$

Malgré les grandes avancées permises par la synthèse $\mathcal{H}_\infty$ dans le domaine de la commande des systèmes aérospatiaux, les correcteurs obtenus sont toujours d'ordre élevé. En effet, le correcteur obtenu $\mathbf{K}(s)$ aura toujours un ordre égal à celui du système augmenté $\mathbf{P}(s)$ ce qui en pratique pose un inconvénient majeur pour pouvoir l'implanter dans les systèmes embarqués. Il est donc nécessaire de réduire l'ordre du correcteur entraînant souvent une baisse de performances du système en boucle fermée.

3.2 Synthèse $\mathcal{H}_\infty$ structurée

En réponse aux inconvénients soulevés dans la section précédente, un nouveau type de synthèse a vu le jour durant la dernière décennie. La synthèse $\mathcal{H}_\infty$ structurée a été développée par les travaux conjoints de [50] et [10] ; elle permet de répondre aux mêmes enjeux que la synthèse $\mathcal{H}_\infty$ tout en imposant une architecture de correcteur *a priori*.

3.2.1 Fonctionnement de la synthèse $\mathcal{H}_\infty$ structurée

La synthèse $\mathcal{H}_\infty$ structurée utilise également la forme standard comme pour la synthèse $\mathcal{H}_\infty$ classique. Cependant elle requiert de définir en plus une structure pour le correcteur, e.g. $\mathbf{K}(\boldsymbol{\lambda}, s) = \text{diag}(\mathbf{K}_1(\boldsymbol{\lambda}, s), ..., \mathbf{K}_n(\boldsymbol{\lambda}, s))$. Chaque élément $\mathbf{K}_i(\boldsymbol{\lambda}, s)$ correspond à la fonction de transfert d'un correcteur de structure prédéfinie par exemple un PID ou une fonction de transfert stable d'ordre quelconque. Ces éléments sont également dépendants des différents

paramètres réglables $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ que l'on cherche à optimiser (gains, coefficients ...).

Il s'agit dans ce cas de trouver un correcteur $\mathbf{K}(\boldsymbol{\lambda}, s)$ stabilisant le système $\mathbf{P}(s)$ et dont la valeur de $\boldsymbol{\lambda}$ minimise la norme $\mathcal{H}_\infty$ de $\mathbf{T_{zw}}$:

$$\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \|\mathbf{T_{zw}}(\mathbf{P(s)}, \mathbf{K}(\boldsymbol{\lambda}, \mathbf{s}))\|_\infty \tag{3.5}$$

3.2.2 Fonctions de pondérations

Chaque contrainte de performance que l'on souhaite imposer au correcteur est présentée sous la forme de fonctions de pondération, ou matrices de fonctions de pondération $\mathbf{W}(s)$. Ce sont celles-ci qui vont permettre de fixer le comportement fréquentiel ou encore optimiser les paramètres des correcteurs. On les formule alors sous la forme de contraintes $\mathcal{H}_\infty$:

$$||\mathbf{W}(s)\mathbf{T}(s)||_\infty < 1 \tag{3.6}$$

avec $\mathbf{T}(s)$, une sous-matrice de transfert de $\mathbf{T_{zw}}(\mathbf{P(s)}, \mathbf{K}(\boldsymbol{\lambda}, \mathbf{s}))$ dont les entrées et sorties sont celles concernées par la pondération $\mathbf{W}(s)$.

Selon le cahier des charges, ces fonctions peuvent prendre des formes très différentes entre un rejet de bruit ou de perturbations ou encore une contrainte en suivi de référence. Dans la plupart des cas, on soumet le système à plusieurs types de contraintes $\mathbf{W}_i(s)$ qui doivent être respectées simultanément, soit :

$$||\mathbf{W}_i(s)\mathbf{T}_i(s)||_\infty < 1,\ i = 1..n \tag{3.7}$$

avec $\mathbf{T}_i(s)$ la matrice de fonctions de transfert entre les entrées exogènes $\mathbf{w}_i$ et les sorties régulées $\mathbf{z}_i$ concernées par la matrice de pondération $\mathbf{W}_i(s)$. Toutes les fonctions de pondérations sont alors regroupées en une contrainte globale $\mathbf{H}(s) = \text{diag}\left(\mathbf{W}_1(s)\mathbf{T}_1(s), ..., \mathbf{W}_n(s)\mathbf{T}_n(s)\right)$ qui devra respecter :

$$||\mathbf{H}(s)||_\infty < 1 \tag{3.8}$$

3.3 Application au co-design

Les études sur la robustesse aux incertitudes ont largement traité de la possibilité d'avoir des correcteurs performants malgré des paramètres variant sur un intervalle de valeurs [8]. Ces études ont abouti à la modélisation LFT qui a permis de séparer le système augmenté $\mathbf{P}(s)$ des paramètres incertains dans un second bloc $\boldsymbol{\Delta}$ (Fig. 2.5).

Dans notre cas, ce bloc $\boldsymbol{\Delta}$ contient les différents paramètres physiques de conception que nous

désirons optimiser regroupés dans le vecteur $\boldsymbol{\delta}$. Celui-ci doit se présenter sous la forme d'une matrice diagonale avec les paramètres $\boldsymbol{\delta}$ apparaissant par nombre d'occurrences (c.-à-d. un paramètre doit être présent autant de fois qu'il apparaît dans le système $\mathbf{P}(s)$).

On peut alors représenter le schéma bloc nécessaire à la réalisation du co-design d'un système à la figure 3.3. On construit alors le correcteur augmenté $\mathbf{C}$ en assemblant le correcteur $\mathbf{K}$ et le bloc $\boldsymbol{\Delta}$ sous un même bloc qui sera donc une matrice diagonale par bloc par construction :

$$\mathbf{C}(\boldsymbol{\lambda}, \boldsymbol{\delta}, s) = \mathrm{diag}(\mathbf{K}(\boldsymbol{\lambda}, s), \boldsymbol{\Delta}(\boldsymbol{\delta})) \tag{3.9}$$

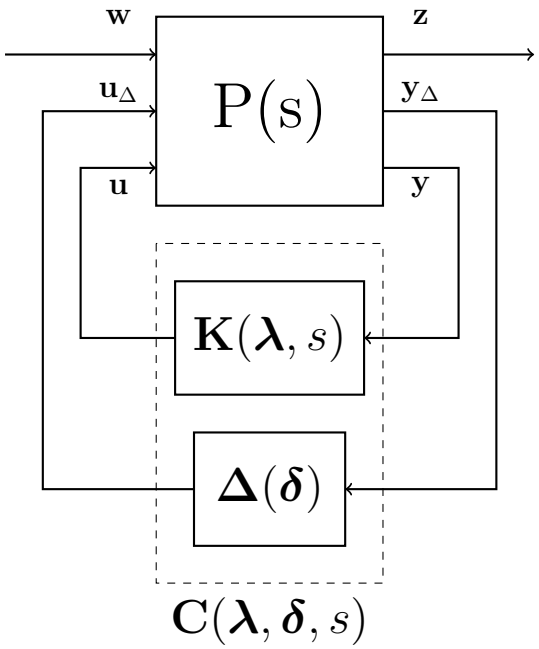

Figure 3.3 Co-design en synthèse $\mathcal{H}_\infty$ structurée

La synthèse $\mathcal{H}_\infty$ structurée revient alors à trouver la valeur *sous-optimale* des paramètres variants $\boldsymbol{\lambda}$ de $\mathbf{K}$ et $\boldsymbol{\delta}$ de $\boldsymbol{\Delta}$ rassemblés dans le correcteur augmenté $\mathbf{C}$, i.e., trouver les valeurs minimisant la norme $\mathcal{H}_\infty$ de $\mathbf{H}(s)$ afin que les contraintes soient respectées.

Afin de bien comprendre comment la synthèse $\mathcal{H}_\infty$ structurée permet d'optimiser les paramètres variants du système, considérons un paramètre scalaire η que l'on cherche à minimiser ou maximiser (par exemple la masse d'un actionneur, le temps de réponse d'un composant, etc.). Le paramètre η dépend des paramètres variants du vecteur $\boldsymbol{\delta}$, leurs valeurs doivent donc être adaptées en conséquence pour rencontrer l'objectif de minimisation ou maximisation. De plus, si l'on souhaite également imposer des contraintes de performances γ au système en boule fermée, il faut choisir des pondérations $\mathbf{W}(s)$ adéquates. Cela va permettre

de concevoir un correcteur $\mathbf{K}$ dont les paramètres variants sont compris dans le vecteur $\boldsymbol{\lambda}$. La synthèse $\mathcal{H}_\infty$ structurée va alors être formulée de la sorte :

$$\begin{aligned} &\min\ \eta \\ &\text{s.t.} ||\mathbf{W_i T_i}(\boldsymbol{\lambda}, \boldsymbol{\delta}, \eta)||_\infty < \gamma,\ i = 1 \dots n,\ \boldsymbol{\delta} \in \boldsymbol{\Delta},\ \boldsymbol{\lambda} \in \boldsymbol{\Lambda} \end{aligned} \tag{3.10}$$

Le plus grand défi de la synthèse $\mathcal{H}_\infty$ structurée est de définir correctement le système et les contraintes qu'on lui impose. C'est dans cette perspective que les algorithmes de MATLAB tels que `hinfstruct` et `systune` (que nous allons utiliser par la suite) ont été conçus, afin de permettre une application simplifiée de la théorie que nous venons de voir.

3.4 Utilisation de `systune`

La fonction `systune` développée par P. Apkarian [10] propose une méthode simple pour la synthèse $\mathcal{H}_\infty$ structurée ; cette fonction est principalement basée sur l'utilisation d'algorithmes d'optimisation non lisse pour régler les gains du correcteur afin de minimiser la norme $\mathcal{H}_\infty$. Le problème est fondamentalement non-convexe, ce qui veut dire que les valeurs initiales données aux paramètres jouent grandement dans l'obtention des valeurs optimales. En effet, il se peut que pour certaines initialisations, l'algorithme ne converge pas vers la solution optimale, voire ne parvienne pas à stabiliser le système.

Ceci est dû à la restriction de l'architecture à une structure définie *a priori* ; cependant il est possible de compenser ce problème de deux façons :

1. Utiliser des conditions initiales aléatoires afin d'exécuter plusieurs fois l'algorithme ;
2. Utiliser des valeurs initiales pour lesquelles le système est déjà stabilisé grâce à une première synthèse du correcteur par une autre méthode.

En pratique, on appliquera souvent ces deux méthodes afin de s'assurer de l'obtention d'un système optimal.

L'utilisation de cette fonction permet d'appliquer la synthèse $\mathcal{H}_\infty$ structurée de façon simple, en particulier dans la définition des critères de pondérations comme nous allons le voir. Cependant, peu de travaux ont cherché à fournir des explications quant à son fonctionnement et son utilisation. Dans le cas particulier de l'application de `systune` au co-design, il est très rare de trouver une documentation à laquelle se fier, ce qui justifie une partie des travaux réalisés dans le cadre de ce mémoire.

Cette section est donc dédiée à détailler comment utiliser `systune` dans le cas d'un système aérospatial pour obtenir les résultats souhaités. Dans un premier temps nous présenterons

comment modéliser le système sous MATLAB puis comment définir les fonctions de pondérations précédemment évoquées.

En Annexe A est également présenté un guide étape par étape de l'utilisation de `systune` pour réaliser la commande en vitesse de rotation d'un moteur et optimiser celui-ci en utilisant le co-design.

3.4.1 Modélisation du système

Avant de définir les différentes contraintes que nous allons imposer au système grâce à `systune` , il est nécessaire de construire le système lui-même en incluant les paramètres variants du correcteur et de la structure à optimiser. Pour cela, il faut créer les points d'analyse qui seront utilisés à cet effet, par exemple les entrées et les sorties du correcteur, ou encore les sorties régulées.

Il est possible d'utiliser deux méthodes pour construire le modèle dans MATLAB .

1 - Construire un modèle `Simulink`

Cette méthode est la plus simple et la plus intuitive, car elle consiste à créer le système sous `Simulink` en nommant les différents blocs et connexions. La fonction `slTuner` permet ensuite d'extraire le système en boucle fermée et les paramètres réglables afin de les utiliser dans la fonction `systune` . Un autre avantage de cette méthode est que le système représenté sous `Simulink` peut ensuite être utilisé pour simuler le fonctionnement du correcteur une fois obtenu afin de vérifier que les contraintes imposées sont bien respectées.

2 - Construire un modèle généralisé sous Matlab

Cette méthode est plus laborieuse, mais permet de décrire plus finement les paramètres réglables (en imposant par exemple un intervalle de valeur), mais aussi de préciser le modèle d'état du système. Contrairement au cas précédent, chaque bloc est défini par des fonctions de transfert en commande MATLAB et les paramètres sous la forme de paramètres `realp` ou de blocs prédéfinis réglables tels que `TunablePID` (pour régler les gains d'un correcteur PID) ou `TunableTF` (pour régler les coefficients d'une fonction de transfert d'ordre quelconque).

Une autre difficulté est de nommer chaque entrée et chaque sortie des fonctions de transfert de telle sorte que celles-ci correspondent aux connexions qui seraient implémentées dans le schéma bloc `Simulink` . Cela est crucial pour pouvoir utiliser la fonction `connect` qui va relier les blocs ainsi définis en associant automatiquement les entrées et les sorties de même

nom et créer des points d'analyse par la même occasion.

Dans la suite du mémoire, nous utiliserons cette méthode. Elle permet plus de flexibilité sur la définition des paramètres et bien qu'elle puisse sembler complexe d'utilisation, le Chapitre 5 et l'Annexe A présentent des exemples d'utilisation afin d'améliorer la compréhension du lecteur.

3.4.2 Définition des contraintes de conception

Une fois le modèle et les paramètres définis sous MATLAB , il faut également écrire les contraintes de performance souhaitées afin de les inclure dans l'algorithme de `systune` . L'un des points forts de cette fonction est précisément qu'elle met à disposition un ensemble de requis de conception grâce à la fonctionnalité `TuningGoal` qui permet de définir de façon simple les critères de pondérations définis plus tôt.

Ces critères prennent la forme de fonctions f dépendant du vecteur $\mathbf{x}$ contenant les paramètres réglables et de la fonction de transfert T entre les entrées exogènes et sorties régulées concernées. Le but de `systune` est alors de faire varier les valeurs de $\mathbf{x}$ de sorte à minimiser f.

Nous ne détaillerons pas ici toutes les fonctionnalités offertes, mais nous présentons plutôt celles utilisées dans la suite du mémoire. Toutefois, une présentation exhaustive de celles-ci et de leurs cas d'application a été effectuée par F. Laliberté dans [51].

Fonction `StepTracking`

Cette première contrainte permet d'imposer au système de suivre la même réponse à un échelon qu'une fonction de transfert de référence $R(s)$ définie au besoin. Ainsi si l'on souhaite par exemple obtenir une réponse d'un système d'ordre 2 standard, la fonction $R(s)$ sera décrite par :

$$R(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2} \tag{3.11}$$

avec ω_n la pulsation propre et ζ le coefficient d'amortissement souhaités. La fonction utilisée par `systune` prend alors la forme donnée ci-dessous.

$$f(\mathbf{x}) = \frac{\|T(s,\mathbf{x}) - \frac{1}{s}R(s)\|_2}{0.1\|\frac{1}{s}R(s) - I\|_2} \tag{3.12}$$

où le coefficient 0.1 est réglable et est nommé `RelGap`. Il permet de déterminer quelle erreur est acceptable entre le système final et le modèle de référence.

En pratique, cette fonction propose de seulement renseigner une constante de temps lorsque la réponse souhaitée est celle d'un ordre 1 ou bien une constante de temps et un dépassement relatif maximal pour un système de référence d'ordre 2.

Fonction `Gain`

Une autre fonction utile de `TuningGoal` est la fonction `Gain`. Elle permet de limiter l'amplitude maximale de la fonction de transfert sur toutes les fréquences ou seulement une bande prédéfinie. La fonction utilisée s'exprime alors sous la forme :

$$f(\mathbf{x}) = \left\| \frac{T(s,\mathbf{x})}{\max_{gain}} \right\|_\infty \tag{3.13}$$

avec $\max_{gain}$ le gain maximal que l'on souhaite imposer.

Une variante de cette fonction est `WeightedGain` qui permet d'imposer des profils de gains en entrée $W_i(s)$ et en sortie $W_o(s)$ sous la forme :

$$f(\mathbf{x}) = \|W_o(s)T(s,\mathbf{x})W_i(s)\|_\infty \tag{3.14}$$

Il est alors possible d'imposer avec une plus grande précision le profil du gain de la fonction de transfert du système en boucle fermée sur différentes bandes de fréquences.

Fonction `Margins`

Une dernière fonction utile dans notre cas est celle qui permet d'imposer des marges de stabilité en gain et en phase au système en boucle fermée. La fonction minimisée par `systune` s'écrit dans ce cas :

$$f(\mathbf{x}) = \|2\alpha S(s,\mathbf{x}) - \alpha I\|_\infty \tag{3.15}$$

où S est la fonction sensibilité de la boucle choisie et α est calculé à partir de la marge de gain minimale GM et marge de phase minimale PM que l'on cherche à imposer selon la formule :

$$\alpha = \max\left(\frac{GM-1}{GM+1}; \tan\left(\frac{PM}{2}\right)\right) \tag{3.16}$$

3.4.3 Exécution de `systune`

La dernière étape est donc d'exécuter l'algorithme de `systune` . Cela requiert de spécifier au minimum deux arguments :

— le modèle en boucle fermée contenant les paramètres réglables `BF` ;
— les contraintes précédemment définies à l'aide de `TuningGoal` rassemblées dans une liste des requis

Cependant, `systune` permet de séparer les différentes contraintes de conception en deux catégories, i.e, les `HardGoals` et les `SoftGoals`. La différence tient dans le fait que les premiers sont ceux que l'algorithme va chercher à minimiser en priorité tandis que les seconds ne seront minimisés que sous couvert que cela n'augmente pas les fonctions associées aux `HardGoals`.

De plus, comme mentionné au début de cette section, `systune` est un algorithme d'optimisation non lisse. Pour compenser ce problème, il est possible d'effectuer plusieurs exécutions de l'algorithme à partir de conditions initiales aléatoires, ce qui est permis dans les options de l'algorithme sous la fonction `systuneOptions('RandomStart',10)` où `10` représente alors ici le nombre d'exécutions à partir de valeurs aléatoires qui sera effectué par `systune` .

Finalement pour effectuer la synthèse $\mathcal{H}_\infty$ structurée, il ne reste qu'à exécuter `systune` en entrant les arguments de la sorte :

```
systune(BF,SoftGoals,HardGoals,Options);
```

CHAPITRE 4 MODELISATION DU SATELLITE

La conception du système de détermination et de commande d'attitude est un des enjeux les plus cruciaux lors du développement d'un satellite. En effet, c'est ce sous-système précis qui permettra d'assurer la stabilité du satellite ainsi que sa capacité à s'orienter une fois celui-ci envoyé dans l'espace, afin par exemple de garantir une grande précision de pointage dans le but de communiquer avec une base située sur le sol terrien ou bien de maintenir une direction fixe pour des instruments optiques. Il est cependant important de remarquer qu'une simple erreur de conception pourrait avoir pour effet de rendre inutilisable le satellite, voire sa chute dans l'atmosphère terrestre.

De nombreuses méthodes de représentation et de commande d'attitude ont été développées et testées, et ce depuis le premier lancement de satellite dans l'espace en 1957. Nous présenterons donc ici dans un premier temps la modélisation cinématique des satellites, puis les différents efforts auxquels ceux-ci peuvent être soumis afin de simuler leur comportement dynamique en incluant les perturbations ainsi que les actionneurs qui permettront de les stabiliser. Une fois le comportement du satellite complètement modélisé, nous verrons comment la commande en attitude est effectuée ainsi que l'architecture à imposer au contrôleur.

4.1 Référentiels

Avant de détailler comment l'orientation du satellite sera modélisée, il semble nécessaire dans un premier temps de définir dans quel référentiel celle-ci est exprimée. Cette section est donc dédiée à la présentation des différents référentiels utilisés pour représenter la cinématique d'un satellite.

4.1.1 Référentiel Inertiel Terrestre

Pour pouvoir définir l'orientation d'un satellite, il nous faut dans un premier temps savoir où celui-ci se situe dans l'espace, et donc quelle orbite est décrite par son mouvement.

Pour une orbite terrestre, le référentiel le plus couramment utilisé est le référentiel ECI *Earth Center Inertial* tel que présenté dans la figure 4.1. C'est dans celui-ci que sont appliquées les lois de Newton afin de décrire les mouvements orbitaux. L'origine de ce repère est le centre de la Terre et ses trois axes sont définis comme suit :

— $\mathbf{z}_i$ est aligné avec le pôle Nord (plus précisément selon l'axe de rotation de la Terre) ;
— $\mathbf{x}_i$ est inclus dans le plan de l'équateur et est dirigé vers l'équinoxe du printemps ;

— $\mathbf{y}_i$ est défini par le produit vectoriel des deux vecteurs précédents.

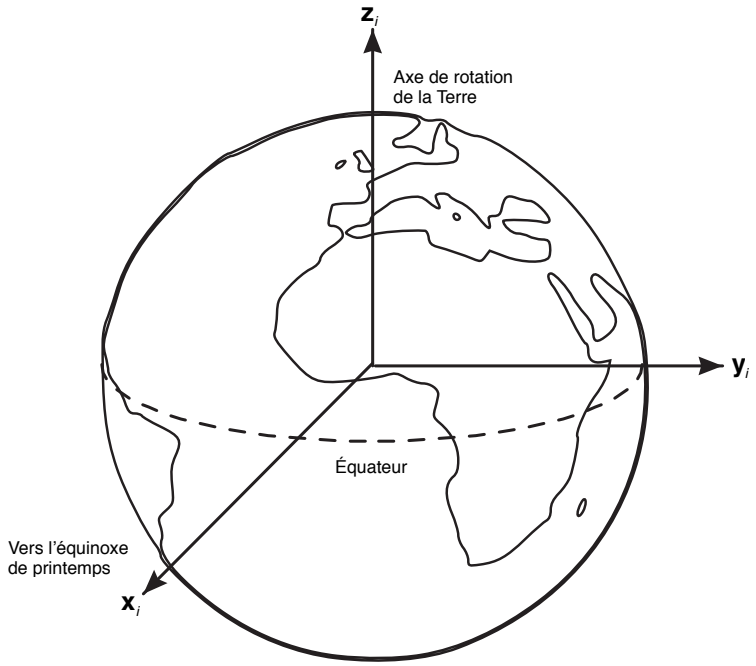

Figure 4.1 Référentiel Inertiel Terrestre (adapté de [2])

4.1.2 Référentiel orbital

Il convient ensuite de définir le référentiel orbital qui représente le mouvement du centre d'inertie du satellite. Ce référentiel suit le satellite au cours de sa rotation autour de la Terre et indique également son sens de rotation. Son origine est donc le centre d'inertie du satellite et ses trois axes sont définis ainsi :

— $\mathbf{z}_o$ est dirigé vers le centre de la Terre ;
— $\mathbf{x}_o$ est inclus dans le plan orbital et est dirigé dans le même sens que la vitesse orbitale ;
— $\mathbf{y}_o$ est défini par le produit vectoriel des deux vecteurs précédents.

En se plaçant au niveau de l'orbite de la Station Spatiale Internationale (ISS - *International Space Station*), on peut décrire complètement les paramètres orbitaux donnés dans le tableau 4.1. Cette orbite est considérée comme circulaire.

Dans la suite du mémoire on utilise cette orbite, car c'est de celle-ci que le Cubesat de *PolyOrbite* devrait être lancé.

Tableau 4.1 Paramètres de l'orbite de l'ISS

Paramètre orbital	Valeur
Inclinaison i (deg)	51.65
Altitude h (km)	400
Vitesse V (km/s)	7.66
Période orbitale (s)	5560

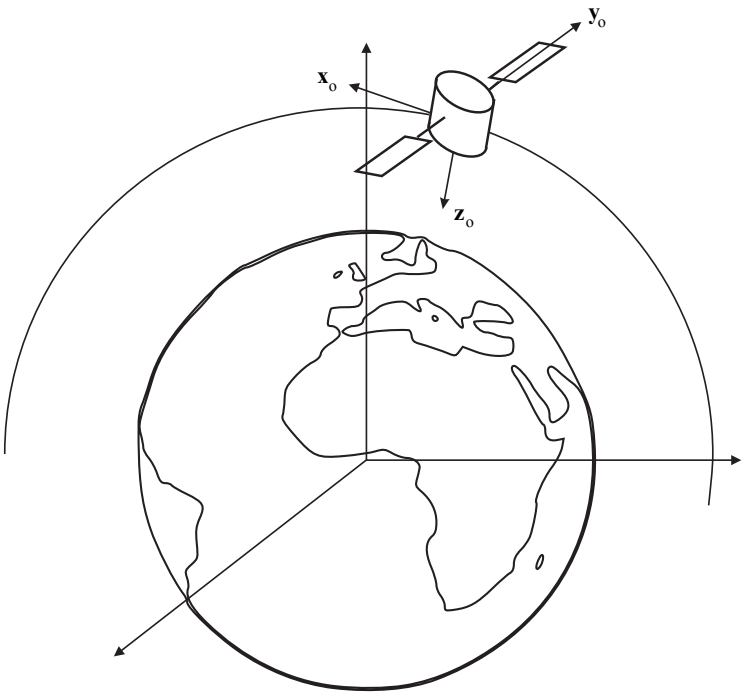

Figure 4.2 Référentiel orbital (adapté de [2])

4.1.3 Référentiel du satellite

Ce référentiel est lié à l'orientation du corps du satellite. Son centre est situé au centre d'inertie du satellite comme le référentiel orbital, mais ses axes sont fixés selon la géométrie de celui-ci. Cela permet, sous l'hypothèse d'un corps rigide, de définir la position de tous les composants du satellite grâce à un vecteur exprimé dans ce repère. Dans la pratique, les axes sont souvent définis selon les axes principaux d'inertie du satellite ou encore dans des directions particulières telles que celle de pointage des systèmes embarqués qui constitue alors l'axe $\mathbf{z}_b$. Il faut cependant que le référentiel soit un système de coordonnées cartésiennes afin de pouvoir représenter aisément l'orientation dans l'espace du satellite.

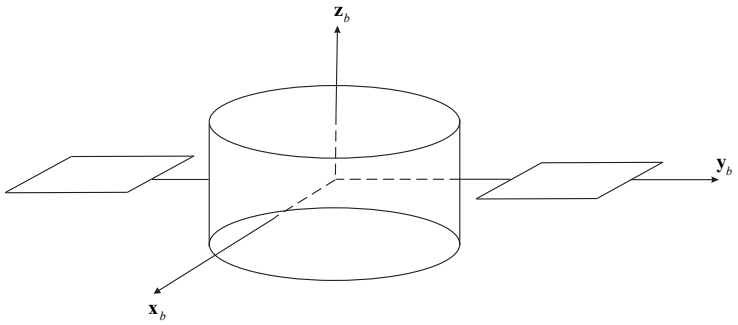

Figure 4.3 Référentiel lié au corps du satellite (adapté de [2])

4.2 Modélisation cinématique

L'attitude d'un satellite est ce qui représente son orientation dans l'espace, et donc l'orientation de ses axes principaux par rapport au référentiel orbital. De nombreuses représentations de cette attitude ont été développées [5], chacune présentant ses avantages et ses inconvénients, mais les plus répandues sont les modélisations par les angles d'Euler et les quaternions (également appelés paramètres d'Euler).

4.2.1 Angles d'Euler

Cette représentation est la plus courante pour décrire l'orientation d'un satellite dans l'espace, elle est en effet la plus intuitive et la plus simple à mettre en œuvre. Elle résulte de trois rotations successives, chacune associée à un angle, ceux-ci étant appelés les angles d'Euler et notés (ϕ, θ, ψ) respectivement appelés angles de roulis, de tangage et de lacet. Ceux-ci sont définis dans la figure 4.4

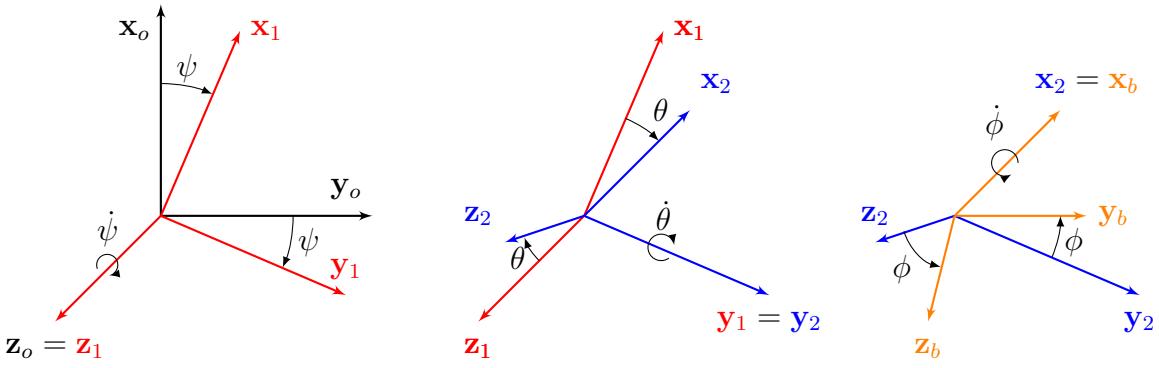

Figure 4.4 Définition des angles d'Euler

Ces angles permettent de représenter l'orientation du référentiel principal du satellite par rapport au référentiel orbital, lui-même en mouvement par rapport au référentiel inertiel terrestre. Chacune des trois rotations est décrite par une matrice de transformation dont les expressions sont détaillées ci-dessous :

1. Dans un premier temps intervient la rotation en lacet ψ autour de l'axe $\mathbf{Z}_o$ du repère inertiel du satellite, dont la matrice de rotation $\mathbf{R}_{\mathbf{z}_o}$ est donnée par :

$$\mathbf{R}_{\mathbf{z}_o}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

2. Ensuite intervient la rotation en tangage θ autour de l'axe $\mathbf{y}_1$ résultant de la rotation précédente comme indiqué dans la figure 4.4 décrite par la matrice $\mathbf{R}_{\mathbf{y}_1}$:

$$\mathbf{R}_{\mathbf{y}_1}(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.2}$$

3. Finalement, une dernière rotation en roulis ϕ autour de l'axe $\mathbf{x}_2$ est donnée par la matrice $\mathbf{R}_{\mathbf{x}_2}$:

$$\mathbf{R}_{\mathbf{x}_2}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{4.3}$$

Ces trois matrices de rotation nous permettent de construire la matrice de passage entre le référentiel principal du satellite et son référentiel inertiel appelée la matrice des cosinus directeurs (**DCM** - *Direct Cosine Matrix*). Celle-ci permet d'obtenir les coordonnées d'un vecteur dans le second référentiel à partir de ses coordonnées dans le premier par une simple multiplication par cette même matrice. On l'obtient comme indiqué dans l'équation 4.4.

$$\mathbf{DCM} = \mathbf{R}_{\mathbf{x}_2}(\phi)\mathbf{R}_{\mathbf{y}_1}(\theta)\mathbf{R}_{\mathbf{z}_o}(\psi) \tag{4.4}$$

En remplaçant les matrices par leurs expressions, on obtient :

$$\mathbf{DCM} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\cos\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{4.5}$$

Comme cette matrice est orthogonale, son inverse est sa transposée et il est ainsi aisé d'obtenir

la matrice de passage du référentiel inertiel au référentiel principal.

Les dérivées des angles d'Euler notés $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ sont connues sous les noms respectifs précession, nutation et rotation. On définit aussi le vecteur instantané de rotation du satellite par rapport au repère orbital par le vecteur $\boldsymbol{\omega} = [p\, q\, r]^\top$. L'équation cinématique d'Euler permet alors de relier les dérivées des angles d'Euler aux composantes du vecteur $\boldsymbol{\omega}$ [32] :

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.6}$$

Cette expression présente une singularité pour $\theta = \pm\dfrac{\pi}{2}$, pouvant mener à des problèmes lors de calculs. Il a donc fallu développer des alternatives à cette solution afin de pallier cet inconvénient.

4.2.2 Quaternions

Une seconde modélisation très répandue est celle des quaternions. Bien que plus difficile à se représenter, elle permet de simplifier les calculs et l'implantation dans les logiciels embarqués et ne présente pas la singularité que l'on trouve avec les angles d'Euler. Également appelés paramètres d'Euler, ceux-ci sont notés q_1, q_2, q_3 et q_4 et sont définis par $\mathbf{q} = [q_1\, q_2\, q_3\, q_4]^\top$ où :

$$\begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^\top = \mathbf{k} \sin\frac{\eta}{2} \text{ et } q_4 = \cos\frac{\eta}{2} \tag{4.7}$$

avec η l'angle de rotation autour de l'axe propre de la rotation du quaternion $\mathbf{q}$ et $\mathbf{k}$ le vecteur unitaire de l'axe de rotation. Par convention, un quaternion $\mathbf{q}$ est souvent pris unitaire, soit :

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{4.8}$$

Une fois que ces quatre paramètres sont connus, il est également possible d'obtenir la matrice de passage **DCM** telle que définie précédemment en utilisant l'expression :

$$\mathbf{DCM} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_1q_4 + q_2q_3) \\ 2(q_1q_3 + q_2q_4) & 2(-q_1q_4 + q_2q_3) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{4.9}$$

De plus, le modèle cinématique est obtenu par la formule exprimant la dérivée du quaternion

$\dot{\mathbf{q}}$ en fonction du quaternion $\mathbf{q}$:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{Q}_{\boldsymbol{\omega}}\mathbf{q} \tag{4.10}$$

avec la matrice $\mathbf{Q}_{\boldsymbol{\omega}}$ définie par :

$$\mathbf{Q}_{\boldsymbol{\omega}} = \begin{bmatrix} 0 & r & -q & p \\ -r & 0 & p & q \\ q & -p & 0 & r \\ -p & -q & -r & 0 \end{bmatrix} \tag{4.11}$$

En développant l'équation 4.10, on obtient le système d'équations cinématiques :

$$\dot{q}_1 = \frac{1}{2}(q_4 p - q_3 q + q_2 r) \tag{4.12}$$

$$\dot{q}_2 = \frac{1}{2}(q_3 p + q_4 q - q_1 r) \tag{4.13}$$

$$\dot{q}_3 = \frac{1}{2}(-q_2 p + q_1 q + q_4 r) \tag{4.14}$$

$$\dot{q}_4 = -\frac{1}{2}(q_1 p + q_2 q + q_3 r) \tag{4.15}$$

Il est important de noter que les représentations sont équivalentes entre elles et que passer des angles d'Euler aux quaternions est possible en utilisant les équations 4.16 telles que présentées ci-dessous :

$$\begin{aligned} \phi &= \arctan\Big(2(q_1 q_4 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)\Big) \\ \theta &= \arcsin\left(2(q_2 q_4 - q_1 q_3)\right) \\ \psi &= \arctan\Big(2(q_3 q_4 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)\Big) \end{aligned} \tag{4.16}$$

4.3 Modélisation dynamique

Les équations de la dynamique pour l'attitude d'un satellite rigide sont décrites par les équations d'Euler et la troisième loi de Newton dans sa forme générale comme indiqué dans [1] :

$$\begin{aligned} \mathbf{J}\dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega} \times \mathbf{H} + \boldsymbol{\tau} \\ \mathbf{H} &= \mathbf{J}\boldsymbol{\omega} + \mathbf{h} \end{aligned} \tag{4.17}$$

où $\boldsymbol{\tau}$ représente le moment selon les 3 axes principaux du satellite, $\mathbf{J} = \mathrm{diag}(J_x, J_y, J_z)$ est la matrice d'inertie du satellite et $\mathbf{h}$ le moment angulaire net induit par la rotation des roues, explicité dans la suite.

Une fois ces équations connues, il est possible de les linéariser autour d'une position d'équilibre $\mathbf{x_e}$. Ceci revient à exprimer le système d'équations sous la forme :

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \tag{4.18}$$

Cela est nécessaire afin de pouvoir synthétiser les correcteurs définis plus loin dans ce chapitre.

Il est possible de diviser les couples $\boldsymbol{\tau}$ en deux catégories : les couples de commande $\boldsymbol{\tau_c}$ qui permettent de stabiliser le satellite et de respecter les contraintes de performance et les couples de perturbation $\boldsymbol{\tau_d}$ qui doivent être compensés par les premiers.

$$\boldsymbol{\tau} = \boldsymbol{\tau_c} + \boldsymbol{\tau_d} \tag{4.19}$$

Il convient à présent de lister les différents types de perturbations et d'actionneurs qui exercent des efforts sur les satellites afin de les modéliser et les incorporer dans les futures simulations.

4.3.1 Perturbations environnementales

Nous allons ici nous concentrer sur les conditions environnementales du satellite et les efforts extérieurs venant perturber le satellite au cours de son orbite. Il est bien entendu nécessaire de prendre en compte celles-ci afin de pouvoir les compenser et de réaliser la commande en orientation. Conformément à l'intuition, celles-ci varient en fonction de l'orbite choisie, par exemple l'altitude et la position relative par rapport à l'orbite lunaire jouent un rôle déterminant dans la définition de l'environnement. Comme nous le verrons, certaines seront parfois négligeables par rapport aux autres comme l'indique la figure 4.5.

Frottement atmosphérique

Même à de grandes altitudes proches du vide spatial, l'atmosphère raréfiée continue d'exercer une pression sur les satellites appelée frottement atmosphérique. Cette force est proportionnelle à la densité de l'atmosphère, qui décroît exponentiellement avec l'altitude conformément à l'intuition, et dépend également de la géométrie du satellite. En effet lorsque la force de pression n'est pas uniformément répartie sur l'aire du satellite, le centre de pression aérodynamique cp_a s'avère être différent du centre de masse cm ; plus la structure est grande et élancée (comme dans le cas de panneaux solaires déployés), plus cette distance peut être

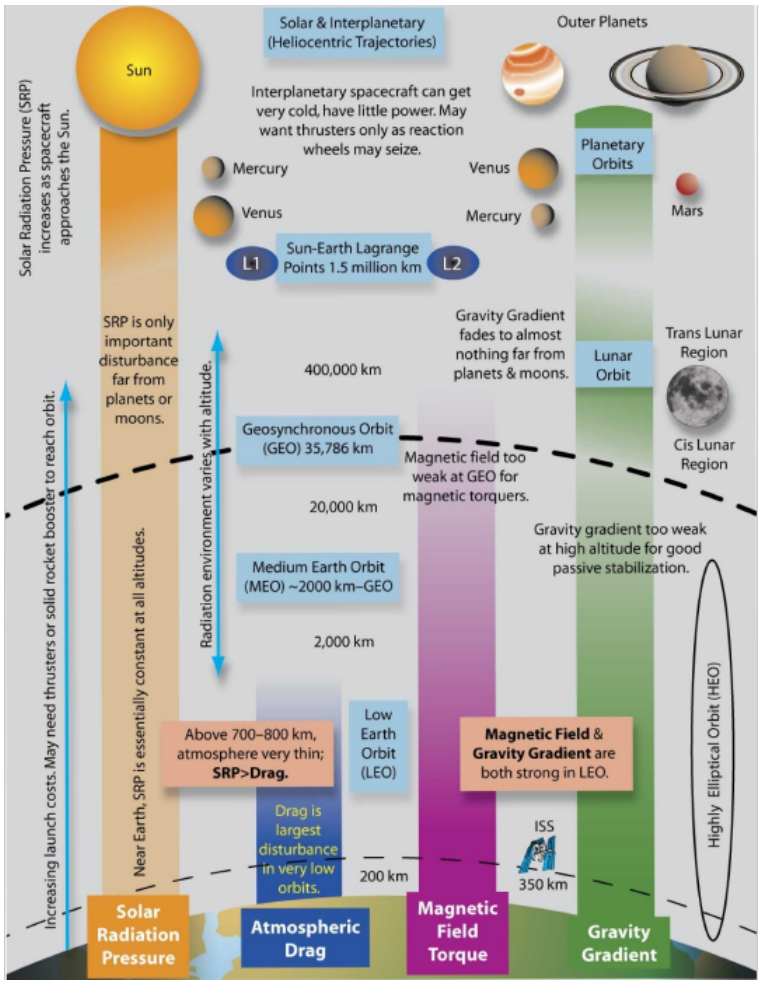

Figure 4.5 Principales perturbations environnementales sur les satellites en orbite terrestre (tiré de [1])

grande et cela induit un couple sur le satellite donné par l'expression :

$$\mathbf{T_{AD}} = \frac{1}{2}\rho C_d A_r \mathbf{V}^2(\mathbf{u_V} \times \mathbf{s_{cp}}) \tag{4.20}$$

où ρ est la densité atmosphérique à l'altitude du satellite ($\rho = 2 \times 10^{-11}$ kg/m^3 pour l'orbite choisie), C_d le coefficient de traînée (souvent compris entre 1 et 2.5), A_r la surface équivalente du satellite dans la direction de mouvement et $\mathbf{V}$ la vitesse orbitale du satellite. Le produit vectoriel est effectué entre $\mathbf{u_V}$ la direction de déplacement dans le référentiel du satellite et $\mathbf{s_{cp}}$ qui est le vecteur joignant le centre de masse cm et le centre de pression aérodynamique cp_a.

Cette perturbation joue un rôle prépondérant pour les basses-orbites, freinant les satellites dans leur rotation et pouvant causer leur entrée dans l'atmosphère. À titre d'exemple, la Sta-

tion Spatiale Internationale (altitude 400 km) doit régulièrement se repositionner en éjectant de grandes quantités de carburants pour compenser le frottement atmosphérique et ne pas s'écraser sur Terre.

Gradient de gravité

En plus de l'atmosphère, la gravité terrestre constitue également une perturbation majeure en basse orbite. Toutefois, celle-ci peut s'avérer être utile en tant que stabilisation passive des satellites lorsque les conditions et missions s'y prêtent et que la précision de pointage n'est pas un facteur important dans la conception du satellite.

Cette perturbation tend à aligner la direction des axes principaux d'inertie vers le centre de la Terre. Dans le cas de structures élancées, cela constitue un véritable défi, car la présence de longs panneaux solaires provoquera inévitablement un couple les orientant vers la Terre et compliquant grandement le maintien en orientation du satellite. Ce couple est calculé de la façon suivante :

$$\mathbf{T_{GD}} = \frac{3\mu}{R^3} \left(\mathbf{u_e} \times (\mathbf{J u_e}) \right) \tag{4.21}$$

avec $\mu = 3.986 \times 10^{14}\,\mathrm{m^3/s^2}$ le paramètre gravitationnel standard de la Terre, R le rayon de l'orbite du satellite ($= 6.778 \times 10^6\,\mathrm{m}$) et $\mathbf{J}$ la matrice d'inertie du satellite. Le vecteur $\mathbf{u_e}$ est quant à lui le vecteur pointant vers le centre de la Terre $\mathbf{z}_o$ exprimé dans le référentiel du satellite.

L'avantage d'utiliser les axes principaux d'inertie du satellite comme repère permet de développer l'équation 4.21 sous la forme :

$$\mathbf{T_{GD}} = \frac{3\mu}{R^3} \begin{bmatrix} (J_z - J_y) \sin\phi \cos\phi \cos^2\theta \\ (J_z - J_x) \sin\theta \cos\theta \cos\phi \\ (J_x - J_y) \sin\psi \cos\theta \sin\theta \end{bmatrix} \tag{4.22}$$

Ensuite, en effectuant l'hypothèse que nous considérons de petits angles autour du point de linéarisation du système et que le satellite est symétrique par rapport à l'axe $\mathbf{z}_b$ et donc que $J_x = J_y$ (comme dans le cas de Cubesats), on peut exprimer (4.21) sous la forme simplifiée :

$$\mathbf{T_{GD}} \approx \frac{3\mu}{R^3} \begin{bmatrix} (J_z - J_y)\phi \\ (J_z - J_x)\theta \\ 0 \end{bmatrix} \tag{4.23}$$

Cette formule bien plus simple fournit une bonne approximation de la perturbation et c'est

celle-ci que nous utiliserons pour synthétiser le contrôleur à partir du modèle linéarisé.

Pression de la radiation solaire

Malgré la distance séparant la Terre de l'astre solaire, celui-ci est la source de perturbations connues sous le nom de pression de radiation solaire. En effet, les particules solaires pourtant nécessaires à la recharge des batteries grâce aux panneaux solaires exercent une pression qui comme dans le cas des frottements atmosphériques est proportionnelle à la distance entre le centre de masse et le centre de pression solaire c_s. La formule qui décrit le couple exercé par cette pression est donnée par l'équation 4.24.

$$\mathbf{T_{SD}} = \mathbf{c_s} \times \mathbf{F_s}, \quad \mathbf{F_s} = (1+K)P_s\mathbf{S}\cos(I_s) \tag{4.24}$$

où $\mathbf{c_s}$ représente le vecteur entre le centre de masse et c_s, K est la réflectance du satellite (estimé à 0.6 pour les petits satellites), $P_s = 4.5 \times 10^{-6}\,\mathrm{N/m^2}$ est la pression solaire effective, $\mathbf{S}$ est le vecteur surface du satellite et enfin I_s l'angle d'incidence entre les rayons du soleil et le vecteur $\mathbf{S}$.

Champ magnétique terrestre

Le champ magnétique terrestre doit également être modélisé, il est la source d'éventuelles perturbations, mais permet avant tout de contrôler l'orientation du satellite par le biais de magnétocoupleurs.

Il est d'usage d'approcher le champ magnétique terrestre par un dipôle magnétique par le biais du modèle IGRF (*International Geomagnetic Reference Field*) à partir de 4 entrées : le rayon de l'orbite r, la colatitude θ , la longitude ϕ et le temps t [52]. Ce modèle propose une expression du potentiel scalaire du champ magnétique en fonction des coefficients numériques de Gauss g et h dont les valeurs sont régulièrement mises à jour [53] :

$$V(r,\theta,\phi,t) = a\sum_{n=1}^{N}\sum_{m=0}^{n}\left(\frac{a}{r}\right)^{n+1}\left[g_n^m(t)\cos(m\phi) + h_n^m(t)\sin(m\phi)\right] \times P_n^m(\cos\theta) \tag{4.25}$$

où $a = 6371.2\,\mathrm{km}$ est le rayon magnétique de référence et P_n^m sont des polynômes de Legendre. Pour le modèle IGRF le développement est arrêté au rang $N = 13$. Le champ magnétique en dérive par la formule du gradient du champ de potentiel :

$$\mathbf{B} = -\nabla V \tag{4.26}$$

Cependant, cette formule est complexe d'utilisation. Il est plus commun d'utiliser un modèle simplifié du champ magnétique tel que donné par [54] :

$$\mathbf{B} = \begin{bmatrix} B_x(t) \\ B_y(t) \\ B_z(t) \end{bmatrix} = \begin{bmatrix} B_0.\cos(\omega_0 t).\sin(i) \\ -B_0\cos(i) \\ 2B_0.\sin(\omega_0 t)\sin(i) \end{bmatrix} \tag{4.27}$$

avec

$$B_0 = \frac{\mu_f}{a^3} \tag{4.28}$$

et où i est l'inclinaison de l'orbite du satellite, a le demi-grand axe de l'orbite, ω_0 la pulsation orbitale et t le temps en considérant $t = 0$ au passage au point ascendant croisant le plan l'équateur. L'intensité du dipôle est donnée par $\mu_f = 7.9 \times 10^{15}\,\text{Wb.m}$. Par exemple, pour l'orbite de l'ISS on a $i = 51.65°$, $a \approx 400\,\text{km}$ et $\omega_0 = 0.0011\,\text{rad/s}$. La figure 4.6 représente la variation du champ magnétique à cette orbite au cours de la révolution du satellite autour de la Terre.

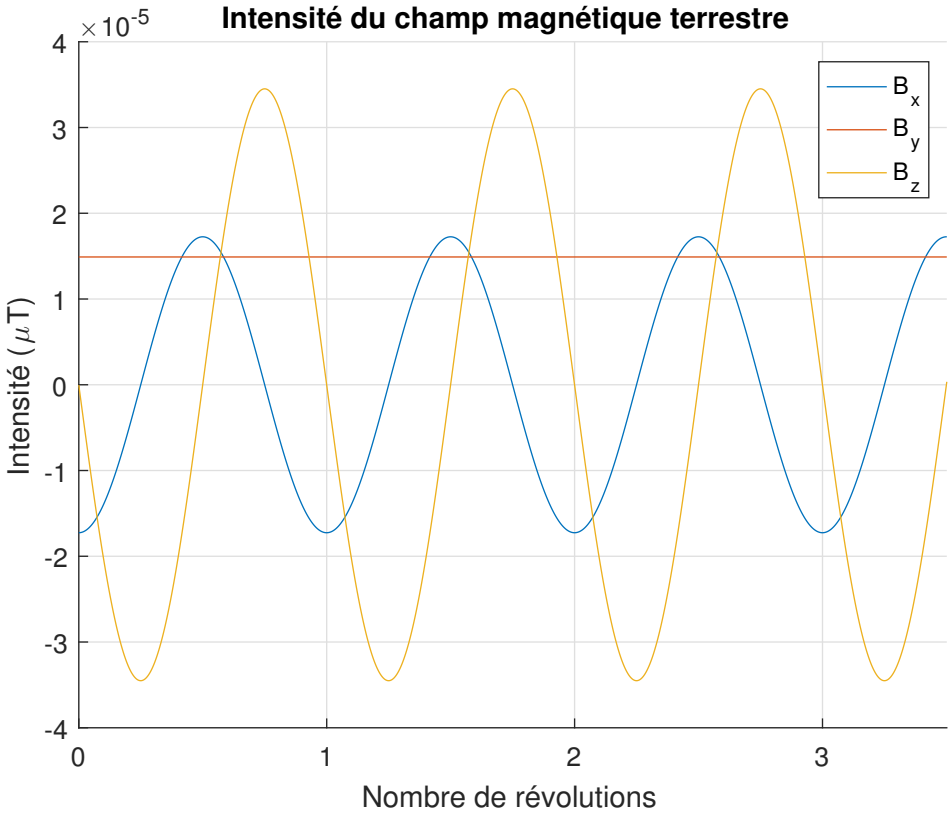

Figure 4.6 Intensité du champ magnétique terrestre

Comme précisé plus tôt, ce champ peut être source de perturbations lorsque les systèmes embarqués exercent un champ magnétique résiduel non souhaité (à cause des magnéto-coupleurs non actionnés, batteries ou antennes). Le moment résiduel magnétique du satellite $\mathbf{m}$ (considéré comme un dipôle) est souvent complexe à estimer, mais pour un nanosatellite des études ont suggéré de considérer $\|\mathbf{m}\| = 0.01\,\mathrm{A.m^2}$ [55]. Le couple résultant peut être estimé par :

$$\mathbf{T_{RM}} = \|\mathbf{m}\|.\mathbf{B} \tag{4.29}$$

Résumé des perturbations extérieures

Dans le cas d'un nanosatellite de type Cubesat situé au niveau de l'orbite de l'ISS, on peut estimer l'intensité des perturbations environnementales. Celles-ci sont regroupées dans le tableau 4.2.

Tableau 4.2 Intensité des perturbations environnementales

Perturbation	Couple exercé (N.m)
Frottement atmosphérique	3.7×10^{-11}
Gradient de gravité	8.4×10^{-10}
Pression solaire	4.3×10^{-9}
Champ magnétique	4.2×10^{-7}

Il faut remarquer ici que le champ magnétique constitue la source principale de perturbations. Cela peut sembler surprenant, mais il faut considérer que le satellite étudié est un nanosatellite et donc de taille très réduite ($\approx 30 \times 10 \times 10\,\mathrm{cm^3}$). Les trois autres sources de perturbations dépendent en effet beaucoup de la taille du satellite ; plus celui-ci est allongé, plus leurs intensités augmentent. On peut donc s'attendre à ce que celles-ci soient bien plus importantes dans le cas d'autres satellites.

4.3.2 Actionneurs

De nombreux types d'actionneurs ont été développés pour assurer la commande en orientation des satellites. Dans ce mémoire nous allons nous concentrer sur deux d'entre eux largement répandus dans les satellites : les roues de réaction et les magnéto-coupleurs. Ceux-ci sont souvent moins lourds et peuvent être utilisés aussi longtemps que les batteries sont rechargées, ce qui en fait des systèmes incontournables pour les nanosatellites.

Roues de réaction

Une des méthodes de commande en orientation les plus connues est le mouvement par échange de moment angulaire : la rotation de cylindres par des rotors à l'intérieur du satellite induit une rotation en sens inverse due à la conservation du moment angulaire total. Ceci est bien entendu possible grâce à la quasi-absence de frottement en orbite autour de la Terre.

Une roue de réaction est constituée d'un moteur entraînant en rotation une roue selon un axe fixe du satellite, tel que représenté à la figure 4.7. Cela produit selon cet axe un couple T qui va augmenter le moment angulaire h de la roue et exercer le moment opposé $-h$ sur le satellite.

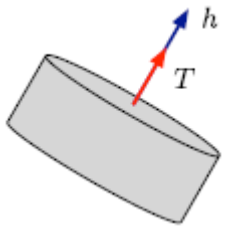

Figure 4.7 Couple d'une roue de réaction (tiré de [3])

On peut lier la variation de moment angulaire $\dot{h}$ d'une roue d'inertie J_w à sa vitesse de rotation Ω et au couple T grâce à l'équation :

$$\dot{h} = J_w \dot{\Omega} = T \tag{4.30}$$

Afin de pouvoir assurer un contrôle complet selon les trois axes du satellite, un minimum de trois roues (fixes) est nécessaire, chacune orientée selon les axes principaux du satellite. Cependant en cas de problème ou panne d'un ou plusieurs des composants, il n'est plus possible de garantir la précision de l'orientation en orbite. Il est ainsi d'usage d'ajouter une roue supplémentaire pour assurer la redondance en cas d'incidents et qui permet bien souvent d'améliorer les performances globales moyennant un algorithme de commande adapté.

En prenant en compte la totalité des roues, l'équation reliant la variation angulaire totale du satellite $\dot{\mathbf{h}}$ aux couples exercés par chaque roue $\mathbf{T}$ devient donc :

$$\dot{\mathbf{h}} = -\mathbf{L}\mathbf{T} \tag{4.31}$$

où $\mathbf{L}$ est la matrice de distribution des roues dans le satellite. Elle a toujours 3 lignes et autant de colonnes que de roues dans le satellite et chaque colonne représente la projection

de la direction des roues dans le repère du satellite. Par exemple dans le cas simple où trois roues sont orientées selon les trois axes principaux, on obtient la matrice identité :

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.32}$$

Une des dispositions des roues les plus répandues est une configuration pyramidale (Fig. 4.8). Chaque roue a ici une contribution comparable au couple total exercé ce qui permet de facilement compenser l'absence d'une roue en cas de panne. Nous allons donc étudier cette configuration dans la suite de ces travaux bien que de nombreuses dispositions soient également possibles (tétraédrique, etc.). Dans ce cas, la matrice de répartition devient :

$$\mathbf{L} = \begin{bmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\beta & -\cos\alpha\cos\beta & \sin\alpha\cos\beta \\ \sin\alpha\cos\beta & \cos\alpha\cos\beta & -\sin\alpha\cos\beta & -\cos\alpha\cos\beta \\ \sin\beta & \sin\beta & \sin\beta & \sin\beta \end{bmatrix} \tag{4.33}$$

avec les angles d'orientation α et d'inclinaison β définis tels que dans la figure 4.9.

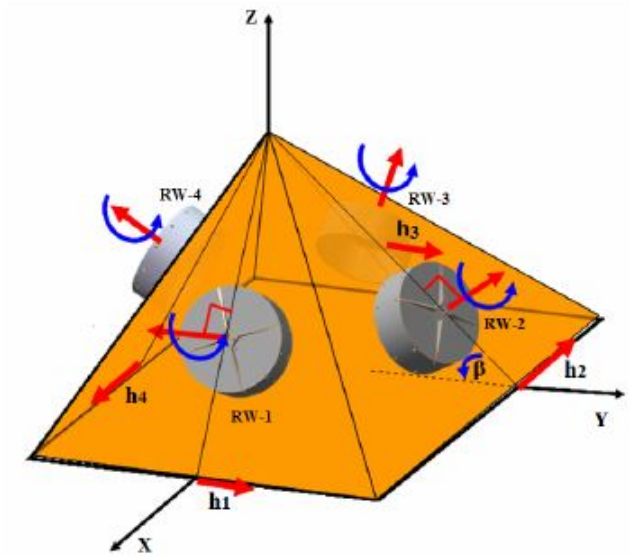

Figure 4.8 Disposition pyramidale des roues de réaction (tiré de [4])

Le couple fourni par l'ensemble des roues de réaction peut alors être exprimé à partir des

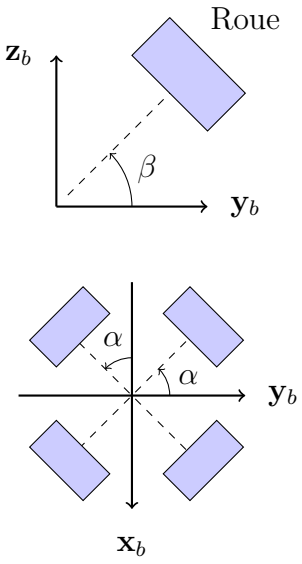

Figure 4.9 Définition des angles α et β

couples exercés individuellement par chaque roue regroupée dans le vecteur $\mathbf{u_w} \in \mathbb{R}^4$:

$$\mathbf{T_{RW}} = \mathbf{L}\mathbf{u_w} \tag{4.34}$$

et de même pour l'accélération angulaire $\dot{\mathbf{\Omega}}$ des roues :

$$\dot{\mathbf{\Omega}} = -J_w^{-1}\mathbf{u_w} - \mathbf{L}^\top \dot{\boldsymbol{\omega}} \tag{4.35}$$

Magnéto-coupleurs

La seconde option classique pour commander l'orientation d'un satellite est d'utiliser le champ magnétique terrestre au travers de bobines appelées magnéto-coupleurs. Selon les principes de l'induction, le fait de faire parcourir un courant dans une bobine elle-même incluse dans le champ magnétique terrestre provoque l'apparition d'un couple au niveau de la bobine. Le fait de faire parcourir un courant i dans une bobine dont les N fils forment une boucle fermée d'aire A produit un moment magnétique m dont l'intensité est donnée par la formule [29] :

$$m = iNA \tag{4.36}$$

Le couple qui est produit par un magnéto-coupleur est alors proportionnel à ce moment $\mathbf{m}$ et au champ magnétique terrestre $\mathbf{B}$ exprimés dans le référentiel du satellite selon l'expression :

$$\mathbf{T_M} = \mathbf{m} \times \mathbf{B} \tag{4.37}$$

Comme pour les roues de réaction, cette technologie nécessite un minimum de trois magnéto-coupleurs, le plus souvent alignés selon les trois axes principaux d'inertie, afin de réaliser le contrôle complet du satellite. Bien que moins précis que ces premières, les magnéto-coupleurs sont plus légers et consomment moins d'énergie ce qui explique qu'ils soient massivement utilisés dans les nanosatellites.

Dans le cas des Cubesats en particulier, la disposition de magnéto-coupleurs la plus répandue est indiquée à la figure 2.3. Le système est composé de deux bobines cylindriques avec un cœur ferromagnétique selon les axes $\mathbf{x}_b$ et $\mathbf{y}_b$ du satellite tandis qu'une bobine de forme carrée est orientée selon l'axe $\mathbf{z}_b$ ce qui permet de réduire au maximum la place occupée dans le nanosatellite.

4.4 Correcteurs

Maintenant que la mécanique de l'orientation des satellites a été définie, nous allons présenter les lois de commande qui permettent d'assurer le fonctionnement et les performances des satellites durant leurs missions.

4.4.1 Architecture de la loi de commande

L'architecture classique des correcteurs de l'orientation des satellites est composée d'un bloc Proportionnel P et d'un bloc Dérivé D [2] dont les gains respectifs seront appelés $\mathbf{K}_p$ et $\mathbf{K}_d$. Ces gains sont réglés de telles sortes que le système respecte les différentes contraintes de performances qu'on lui impose. On distingue deux catégories principales de contraintes : les contraintes de performances en boucle fermée telles que le temps de réponse ou bien l'amortissement des dépassements et les contraintes en marge de gains et de phase.

Angles d'Euler

La plupart des correcteurs sont formés par ce que l'on appelle des retours de sortie : des grandeurs physiques sont mesurées puis comparées aux valeurs de référence. Le contrôleur utilise alors l'écart entre celles-ci pour commander le système et atteindre les valeurs souhaitées.

Dans notre cas, la référence est l'attitude désirée du satellite, i.e., des angles désirés de roulis,

tangage et lacet $(\phi_c, \theta_c, \psi_c)$. Le système en boucle fermée permet donc de calculer l'écart $\mathbf{e}$ entre attitude réelle et de référence par la formule :

$$\mathbf{e} = \begin{bmatrix} \phi_c - \phi \\ \theta_c - \theta \\ \psi_c - \psi \end{bmatrix} \tag{4.38}$$

Dans ce cas, le calcul du couple de commandes se fera par l'expression [5] :

$$\mathbf{T_c} = \mathbf{K}_p\mathbf{e} + \mathbf{K}_d\boldsymbol{\omega} \tag{4.39}$$

Quaternions

Lorsque la modélisation par les quaternions est choisie pour concevoir l'algorithme de commande, il est nécessaire de définir le quaternion erreur $\mathbf{q}_e$ tel que décrit dans la formule ci-dessous :

$$\mathbf{q}_e = \begin{bmatrix} q_{e,1} \\ q_{e,2} \\ q_{e,3} \\ q_{e,4} \end{bmatrix} = \begin{bmatrix} q_{c,4} & q_{c,3} & -q_{c,2} & -q_{c,1} \\ -q_{c,3} & q_{c,4} & q_{c,1} & -q_{c,2} \\ q_{c,2} & -q_{1,c} & q_{c,4} & q_{c,3} \\ q_{1,c} & q_{c,2} & q_{c,3} & q_{c,4} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{4.40}$$

avec $\mathbf{q}_c = \begin{bmatrix} q_{c,1} & q_{c,2} & q_{c,3} & q_{c,4} \end{bmatrix}^\top$ le quaternion désiré, représentant l'attitude que l'on souhaite imposer au satellite. Le couple de commande est dans ce cas [4] :

$$\mathbf{T_c} = -\mathbf{K}_p\bar{\mathbf{q}}_e - \mathbf{K}_d\boldsymbol{\omega} \tag{4.41}$$

avec $\bar{\mathbf{q}}_e = \begin{bmatrix} q_{e,1} & q_{e,2} & q_{e,3} \end{bmatrix}^\top$. Comme nous considérons des matrices de gains diagonales, nous pouvons développer (4.41) pour obtenir :

$$\begin{aligned} T_{c,x} &= -K_{p_1} q_{e1} - K_{d_1} p \\ T_{c,y} &= -K_{p_2} q_{e2} - K_{d_2} q \\ T_{c,z} &= -K_{p_3} q_{e3} - K_{d_3} r \end{aligned} \tag{4.42}$$

4.4.2 Commande des roues de réaction

Nous venons de voir comment un contrôleur PD permettait d'imposer une dynamique et des performances à un satellite en comparant des mesures physiques à la commande souhaitée puis en calculant le couple nécessaire à d'éventuelles corrections d'attitude. Il est à

présent nécessaire de détailler comment ces couples de commandes sont ensuite inclus dans la modélisation du satellite.

Le sujet de la répartition du couple de commandes entre les différentes roues est traité dans [56]. En effet en utilisant la matrice de distribution $\mathbf{L}$, il est possible de calculer le couple $\mathbf{u_w}$ que chaque roue doit fournir afin que le satellite reçoive l'effort calculé par le correcteur :

$$\mathbf{u_w} = \mathbf{L}^+(\boldsymbol{\omega} \times \mathbf{H} + \mathbf{T_c}) \tag{4.43}$$

où $\mathbf{L}^+$ est la pseudo-inverse de la matrice $\mathbf{L}$ obtenue par la formule :

$$\mathbf{L}^+ = \mathbf{L}^\top(\mathbf{L}\mathbf{L}^\top)^{-1} \tag{4.44}$$

Il est important de remarquer ici que la loi de commande n'est pas un contrôleur PD simple standard à cause du terme de produit vectoriel ; celui-ci permet de compenser les effets gyroscopiques induits par la rotation des roues. L'architecture de commande du satellite lors du mode de pointage est alors représenté dans la figure (4.10).

4.4.3 Désaturation des roues de réaction

La désaturation d'une roue consiste à réduire la vitesse de rotation d'une ou plusieurs roues une fois que celles-ci ont atteint leur vitesse maximale, les empêchant ainsi de pouvoir fournir davantage de couple. Cependant, réduire la vitesse de rotation des roues provoque un couple non souhaité sur le satellite qui doit conserver son attitude.

Selon [57], il faut donc fournir au satellite un moment angulaire de valeur opposée à celui induit par la décélération des roues. Le moment angulaire à produire dans le référentiel du satellite est donné par :

$$\boldsymbol{\Delta h} = \mathbf{L}(\mathbf{h_w} - \mathbf{h_{ref}}) \tag{4.45}$$

où $\mathbf{h_w}$ correspond au moment angulaire de chaque roue à un instant t.

Pour compenser $\boldsymbol{\Delta h}$, les magnéto-coupleurs doivent donc suivre la relation ci-dessous tout au long de la décélération :

$$\mathbf{m} = -\frac{\mathbf{K_{mag}}}{\|\mathbf{B}\|^2}(\mathbf{B} \times \boldsymbol{\Delta h}) \tag{4.46}$$

Cette loi de commande s'avère efficace même si le temps nécessaire pour ralentir les roues est souvent de l'ordre de plusieurs orbites du satellite. Quelques études ont été consacrées à l'étude de ce mode de fonctionnement particulier des satellites : [58] a par exemple cherché

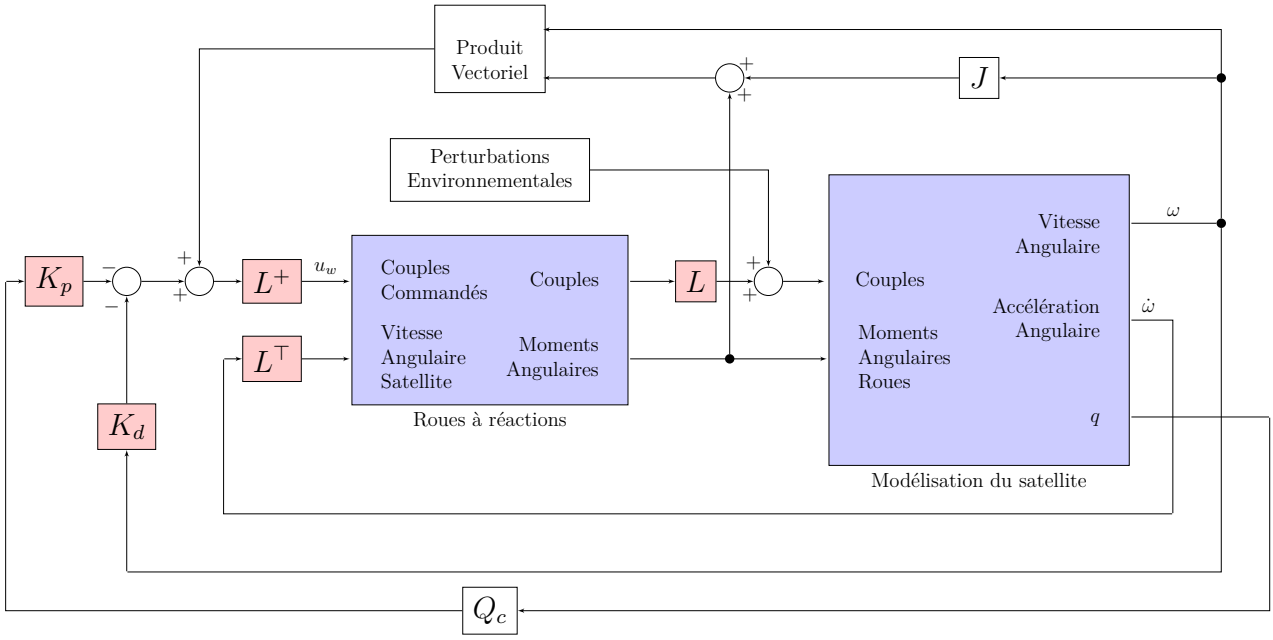

Figure 4.10 Système de commande par roues de réaction (quaternions)

à trouver le système optimal permettant de réduire au possible la consommation en énergie nécessaire à cette opération.

CHAPITRE 5 APPLICATION DE SYSTUNE A L'OPTIMISATION DE SYSTÈMES

Ce chapitre rassemble les travaux réalisés en collaboration avec P. Lallet, étudiant de maîtrise professionnelle et qui ont été présentés à la conférence *AIAA SPACE 2018* à Orlando dans la catégorie *Student Paper Competition* [59].

L'objectif principal de ce chapitre est d'optimiser le système d'orientation composé de roues de réaction tout en respectant les contraintes de performances imposées et certaines restrictions géométriques qui seront détaillées plus tard. Afin de réaliser cela, nous allons utiliser la synthèse $\mathcal{H}_\infty$ structurée à l'aide de MATLAB et de la fonction `systune`, ce pour quoi il est nécessaire de paramétrer notre système.

5.1 Définition du système

5.1.1 Paramétrisation des actionneurs

Afin de pouvoir utiliser des paramètres variables avec l'algorithme `systune`, il faut les définir en tant que fonction de coefficients `realp`. Dans le contexte de notre étude, il s'agira de l'inertie des quatre roues de réaction J_w et la matrice de distribution $\mathbf{L}$.

Inertie des roues de réaction

La valeur de départ pour la simulation de l'inertie des roues sera fixée à $0.0077\,\mathrm{kg.m^2}$, mais dans un objectif d'optimisation des performances nous allons permettre à la valeur de J_w de varier dans un intervalle prédéfini de valeurs. La variable J_w est donc défini dans MATLAB en tant que paramètre `realp` dont les valeurs minimum et maximum seront imposées de telle sorte à respecter la contrainte d'espace disponible. Dans cette simulation la valeur maximale est fixée à $0.01\,\mathrm{kg.m^2}$ et la valeur minimale à $0.0001\,\mathrm{kg.m^2}$.

Matrice de distribution L

D'après [4], la matrice de distribution pour ce type de configuration peut être définie à l'aide des deux angles α et β, selon l'équation 5.1

$$\mathbf{L} = \begin{bmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\beta & -\cos\alpha\cos\beta & \sin\alpha\cos\beta \\ \sin\alpha\cos\beta & \cos\alpha\cos\beta & -\sin\alpha\cos\beta & -\cos\alpha\cos\beta \\ \sin\beta & \sin\beta & \sin\beta & \sin\beta \end{bmatrix} \tag{5.1}$$

Cependant, il n'est pas possible de définir telle quelle la matrice **L** dans MATLAB avec α et β en tant que coefficients `realp` car ceux-ci ne peuvent être utilisés en arguments des fonctions cosinus et sinus.

Il est donc nécessaire de transformer l'expression de **L** afin que celle-ci ne soit composée que d'opérations simples comme l'addition, soustraction, multiplication ou la division entre les différents coefficients. Cela peut être réalisé en utilisant les formules de la tangente de l'arc moitié :

$$\begin{aligned} \cos(x) &= \frac{1-t^2}{1+t^2} \\ \sin(x) &= \frac{2t}{1+t^2} \end{aligned} \tag{5.2}$$

avec $t = \tan\left(\frac{x}{2}\right)$. Nous prenons donc pour paramètres $a = \tan\left(\frac{\alpha}{2}\right)$ et $b = \tan\left(\frac{\beta}{2}\right)$ ce qui nous permet de réécrire la matrice de distribution sous la forme :

$$\mathbf{L} = \begin{bmatrix} \frac{(1-a^2)(1-b^2)}{(1+a^2)(1+b^2)} & -\frac{2a(1-b^2)}{(1+a^2)(1+b^2)} & -\frac{(1-a^2)(1-b^2)}{(1+a^2)(1+b^2)} & \frac{2a(1-b^2)}{(1+a^2)(1+b^2)} \\ \frac{2a(1-b^2)}{(1+a^2)(1+b^2)} & \frac{(1-a^2)(1-b^2)}{(1+a^2)(1+b^2)} & -\frac{2a(1-b^2)}{(1+a^2)(1+b^2)} & -\frac{(1-a^2)(1-b^2)}{(1+a^2)(1+b^2)} \\ \frac{2b}{1+b^2} & \frac{2b}{1+b^2} & \frac{2b}{1+b^2} & \frac{2b}{1+b^2} \end{bmatrix} \tag{5.3}$$

Cette expression permet bien d'avoir chaque coefficient écrit sous la forme de fractions rationnelles, fonction des paramètres a et b. Les valeurs de a et b sont contenues entre 0 et 1 car elles correspondent à des angles de 0 et π radians. On peut obtenir les valeurs de α et β en calculant simplement $\alpha = 2\arctan a$ et $\beta = 2\arctan b$.

Afin de résumer les intervalles de définition de chaque paramètre, on les regroupe dans le tableau 5.1.

Tableau 5.1 Définition des paramètres

Paramètre	Valeur initiale	Minimum	Maximum
J_w (kg.m^2)	0.0077	0.0001	0.01
a	0	0	1
b	$\tan\left(\frac{\pi}{4}\right)$	0	1

5.1.2 Exigences de conception

Notre correcteur doit respecter plusieurs contraintes telles que le temps de réponse du système ou encore limiter l'effort des actionneurs, tout en réduisant la consommation d'énergie et en assurant la stabilité du système en boucle fermée. Nous allons donc appliquer les contraintes suivantes à notre système.

Réponse à un échelon

On souhaite contraindre le système à avoir un temps de réponse T_r environ égal à 20 s pour une commande en échelon sur chaque axe principal du satellite, avec un dépassement aussi faible que possible et sans erreur en régime permanent. La fonction de transfert de référence permettant de respecter ces contraintes est une fonction d'ordre 2 qui se présente sous la forme :

$$H_{ref} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2} \tag{5.4}$$

où ζ sera augmenté autant que possible afin de réduire le dépassement sous la contrainte du temps de réponse T_r. Cette contrainte est une des plus importantes car elle impose la dynamique du satellite. Elle sera donc définie en tant que `HardGoal` dans `systune` pour s'assurer de son respect avant toutes les autres contraintes de conception.

Limitation du couple de commande

On souhaite également imposer une valeur maximale pour le couple des roues de réaction afin de respecter les limites physiques du système. Dans notre cas, cette valeur est fixée à 0.5 N.m. Ceci est imposé au système par le biais d'un gain maximum à la fonction de transfert entre la commande en orientation et le couple des roues de réaction.

Réduction de la consommation en énergie

Nous cherchons également à optimiser le satellite sous la contrainte de réduction de la consommation d'énergie des roues de réaction. Pour cela, nous allons faire varier les différents paramètres que nous avons définis précédemment et imposer un gain maximum sur la fonction de transfert entre la commande et la vitesse de rotation des roues afin de limiter par la même occasion la consommation d'énergie.

5.2 Utilisation de `systune`

5.2.1 Paramètres variables

Il nous faut dans un premier temps définir les paramètres que nous allons faire varier au cours de l'optimisation avec `systune` . Comme vu dans les sections précédentes, cela est effectué en définissant des paramètres `realp` à partir d'une valeur initiale. Il est ensuite possible de définir un intervalle pour restreindre les valeurs possibles des paramètres à un domaine physiquement envisageable.

Nous pouvons donc définir l'inertie des roues J_w en tant que paramètre variable tel que décrit ci-dessous :

```
Jw=realp('Jw',0.0077);
Jw.Minimum=0;
Jw.Maximum=0.01;
```

De même, nous pouvons définir l'angle d'inclinaison β, qui est représenté par le coefficient $b = \tan\left(\frac{\beta}{2}\right)$ comme nous l'avons vu précédemment :

```
b           = realp('b',tan(pi/8));
b.Minimum   = 0;
b.Maximum   = 1;
```

On définirait de la même façon le paramètre $a = \tan\left(\frac{\alpha}{2}\right)$.

Le but étant de concevoir le système de commande en parallèle de l'optimisation des actionneurs, il nous faut également définir les gains du correcteur PD en tant que variables d'optimisation. Nous allons par ailleurs leur imposer une structure de matrice diagonale en imposant une valeur nulle aux coefficients non diagonaux par la formule `.free(m,n)=false` où `m` et `n` sont respectivement les indices de lignes et de colonnes.

```
Kp              = realp('Kp',eye(3));
Kp.Minimum      = zeros(3);
Kp.Maximum      = 2*eye(3);
Kp.free(1,2)    = false;
Kp.free(1,3)    = false;
Kp.free(2,1)    = false;
Kp.free(2,3)    = false;
Kp.free(3,1)    = false;
Kp.free(3,2)    = false;
```

La matrice du gain Dérivé $\mathbf{K}_d$ est définie de façon similaire.

5.2.2 Définition du système

Une fois les paramètres définis, il faut les connecter à la modélisation globale du satellite. Cependant, comme nous avons pu le voir dans le Chapitre 4 le système est non linéaire ; il est donc nécessaire de la linéariser afin de pouvoir utiliser `systune` . À des fins pratiques, nous avons donc utilisé la fonction `linmod` de MATLAB pour obtenir ce modèle linéaire à partir du schéma Simulink de la figure 5.1 où sont implémentés les équations présentées précédemment. On obtient ainsi le modèle $\mathbf{G}$ où les entrées et les sorties sont définies à la figure 5.2.

Il faut ensuite modéliser tous les blocs et paramètres en tant que fonctions de transfert afin de pouvoir les relier au système linéarisé. Cette partie peut être source d'erreurs, car chaque entrée et sortie doivent être nommées avec précision afin que les liaisons soient reconnues dans l'algorithme. Pour cela on utilise donc les fonctions `InputName` et `OutputName`, comme réalisé ci-dessous pour le modèle linéarisé :

```
[A,B,C,D]       = linmod('linsat');
G               = ss(A,B,C,D);
G.InputName     = {'Mx','My','Mz','hx','hy','hz'};
G.OutputName    = {'phi','theta','psi','omegax','omegay','omegaz',
                   'omega_dx','omega_dy','omega_dz'};
```

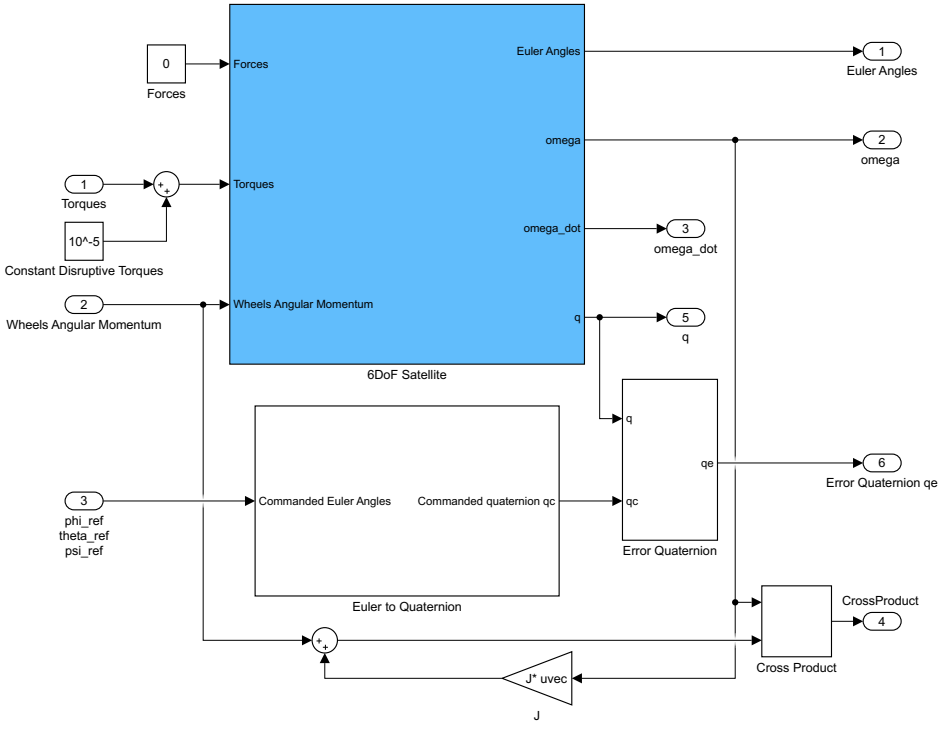

Figure 5.1 Schéma SIMULINK du modèle non linéaire du satellite

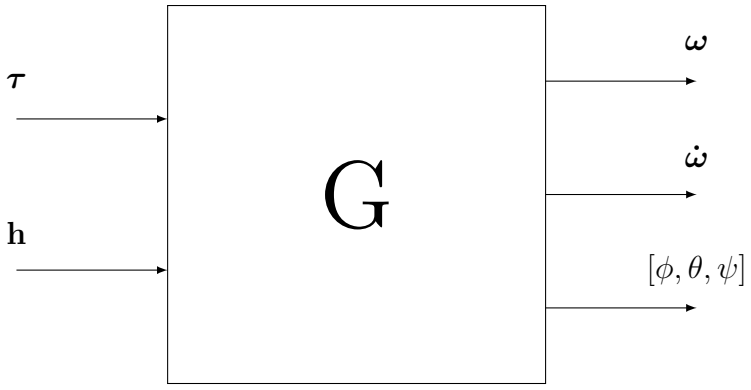

Figure 5.2 Linéarisation du satellite

Les paramètres doivent être compris dans un bloc extérieur au modèle linéarisé afin de pouvoir optimiser leurs valeurs. On construit la matrice **L** (Eq. 5.1) à partir des angles d'inclinaison α et β (représentés par a et b) précédemment définis comme paramètres `realp` :

```
ca  = (1-a^2)/(1+a^2);
sa  = 2*a/(1+a^2);
cb  = (1-b^2)/(1+b^2);
sb  = 2*b/(1+b^2);
L   = [ca*cb -sa*cb -ca*cb  sa*cb
       sa*cb  ca*cb -sa*cb -ca*cb
       sb     sb     sb     sb ]
```

De même, le bloc proportionnel du correcteur est défini tel que suit :

```
Kp_transf = tf(Kp*eye(3));
Kp_transf.u = {'phie','thetae','psie'};
Kp_transf.y = {'Kp_out1','Kp_out2','Kp_out3'};
```

Il en va de même pour le bloc $\mathbf{K}_d$.

Pour introduire une somme, nous allons utiliser la fonction `sumblock`, un exemple d'implantation est présenté ci-dessous pour le calcul du moment commandé aux roues d'inertie selon l'axe $\mathbf{x}_b$ du satellite :

```
sum1 = sumblk('Mx=-Kp_out1-Kd_out1');
```

Une fois que toutes les fonctions de transfert, entrées et sorties du système ont été implémentées dans MATLAB (en incluant le correcteur) nous allons pouvoir utiliser la fonction `connect`. Cette fonction prend en argument toutes les fonctions de transfert préalablement définies ainsi que les entrées et sorties du système en boucle fermée. Dans notre cas, les entrées seront les angles d'orientation souhaités du satellite et les sorties les quaternions et les angles représentant l'attitude du satellite, les vitesses angulaires du satellite ainsi que leurs dérivées, les vitesses de rotation des 4 roues d'inerties et enfin les sorties associées à la matrice inerties dans le but d'optimiser celle-ci.

```
sys=connect(Kp_transf,Kd_transf,Jw_inv,Lpf,Ltf,LJf,Jwf,G,int1,sum1
    ,sum2,sum3,sum4,sum5,sum6,sum7,{'phi_ref','theta_ref','psi_ref'
    },{'q1','q2','q3','q4','phi','theta','psi','omegax','omegay','
    omegaz','omega_dx','omega_dy','omega_dz','Omega1','Omega2','
    Omega3','Omega4','Jwf_out1','Jwf_out2','Jwf_out3','Jwf_out4'});
```

5.2.3 Définition des contraintes

Il y a un vaste panel de possibilités de contraintes dans l'interface de `systune`, celles-ci sont réunies dans le bloc `TuningGoal`. Une présentation de quelques-unes d'entre elles a été fournie dans le chapitre précédent.
Pour la contrainte de temps de réponse, nous utilisons la fonction appelée `StepTracking`. Celle-ci permet d'imposer le temps de réponse ainsi que le dépassement maximal de la réponse temporelle.

```
TrackReq = TuningGoal.StepTracking({'phi_ref','theta_ref','psi_ref
    '},{'phi','theta','psi'},tau_response,overshoot);
```

Il est également possible de contraindre l'algorithme à limiter le couple maximal produit par chaque roue en imposant un gain maximum à la fonction de transfert correspondante en utilisant la fonction `TuningGoal.Gain` :

```
CommandTorqueReq = TuningGoal.Gain({'phi_ref','theta_ref','psi_ref
    '},{'Jwf_out1','Jwf_out2','Jwf_out3','Jwf_out4'},max_torque);
```

Il en va de même pour la limitation de la vitesse de rotation des roues et de leur accélération angulaire afin de respecter les limitations physiques des moteurs :

```
CommandSpeedReq = TuningGoal.Gain({'phi_ref','theta_ref','psi_ref'
    },{'Omega1','Omega2','Omega3','Omega4'},max_speed);

Omegad_Req = TuningGoal.Gain({'phi_ref','theta_ref','psi_ref'},{'
    Omegad_1','Omegad_2','Omegad_3','Omegad_4'},1);
```

Enfin, une fois que toutes les contraintes de conception sont définies, il nous reste à les classer selon les catégories `SoftGoals` et `HardGoals`, dont la différence a déjà été abordée dans le chapitre précédent.

```
SoftReqs = [Omegad_Req];
HardReqs = [TrackReq,CommandTorqueReq,CommandSpeedReq];
```

5.2.4 Systune

Il est alors possible d'exécuter la fonction `systune` dans MATLAB :

```
ST = systune(sys_approx,SoftReqs,HardReqs);
```

Toutefois, afin d'améliorer les résultats il est souvent nécessaire d'effectuer plusieurs exécutions de `systune` à partir de valeurs aléatoires. On utilise donc à cet effet la fonction `systuneOptions`. En imposant la valeur 100 dans l'exemple ci-dessous, on impose à l'algorithme de s'exécuter autant de fois à partir de valeurs différentes ce qui augmente la chance d'approcher le minimum global et donc la combinaison optimale des paramètres.

```
op  = 100;
opt = systuneOptions('RandomStart',op);
ST  = systune(sys_approx,SoftReqs,HardReqs,opt);
```

5.3 Simulations

Cette section est dédiée à l'application de la synthèse du correcteur telle que décrite précédemment dans le cadre d'un satellite à l'aide de MATLAB et `Simulink` . Nous allons donc déterminer ici dans quelle mesure nous pouvons effectivement réduire la consommation en énergie, tout en respectant les contraintes de performance imposées. Afin de voir également l'impact des paramètres dans l'optimisation, nous allons étudier au fur et à mesure l'ajout de plusieurs grandeurs variantes à savoir l'angle d'inclinaison des roues β, les gains du correcteur $\mathbf{K}_p$ et $\mathbf{K}_d$ et enfin l'inertie des roues de réaction J_w.

5.3.1 Application à un cas de satellite

Le satellite considéré ici est un microsatellite qui a été étudié dans [4]. Ses caractéristiques, ainsi que les conditions initiales de la simulation sont données dans le tableau 5.2. Les valeurs initiales des gains du correcteur ont été choisies d'après [4] afin de garantir la stabilité du système et un temps de réponse déjà satisfaisant.

5.3.2 Résultats

Optimisation de l'angle d'inclinaison β

Dans un premier temps, seul l'angle d'inclinaison est défini comme variable dans la simulation. Les autres valeurs restent fixées à celles données dans le Tableau 5.2.

Une fois la simulation exécutée, `systune` retourne un angle $\beta = 35.27°$. Celui-ci demeure constant même lorsque le nombre de `RandomStarts` est augmenté et est très proche de la valeur obtenue par Abdellatif Bellar dans [4]. Ceci permet déjà de confirmer que la synthèse

Tableau 5.2 Conditions initiales pour le co-design

Paramètre	Valeur	Unité
Matrice d'inertie du satellite $\mathbf{J}$	$\begin{bmatrix} 12 & 0 & 0 \\ 0 & 14 & 0 \\ 0 & 0 & 10 \end{bmatrix}$	kg.m^2
Inertie des roues de réaction J_w	77.10^{-4}	kg.m^2
α	0	°
β	45	°
$\mathbf{K}_p$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
$\mathbf{K}_d$	$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$	
Attitude initiale	$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$	°
Rotation angulaire initiale ω	$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$	rad/s
Attitude commandée	$\begin{bmatrix} 30 & 20 & 40 \end{bmatrix}^\top$	°

avec `systune` peut bel et bien s'appliquer à des fins de co-design.

Nous pouvons ensuite vérifier que les contraintes imposées au système sont bien respectées sur les réponses en commandes des angles d'Euler et sur les couples et vitesses de rotation des roues. De plus, la puissance consommée par les roues est représentée sur les figures 5.3 et 5.4. Les temps de réponse paraissent bien être compris entre 20 et 30 s pour passer de l'attitude initiale à l'attitude finale comme cela était souhaité ; malgré de légers dépassements, les requis sont tous respectés dans leur ensemble.

Comparaison au système initial

Dans la conception initiale du satellite l'angle d'inclinaison avait pour valeur $\beta = 45°$. Il semble donc naturel de comparer les performances avec cette première optimisation offerte par l'algorithme. La figure 5.5 nous permet de constater en effet un léger écart entre les consommations des deux versions du système, bien que l'écart soit faible.

Bien qu'il soit apparent que les performances de notre système aient bien été améliorées un peu en optimisant un unique paramètre, cela peut être fait aisément par d'autres méthodes sans nécessairement recourir à `systune` . L'intérêt de cette fonction pour le co-design se révèle avant tout lorsque l'on cherche à optimiser de nombreux paramètres simultanément, ce qui est l'objet du reste de cette section.

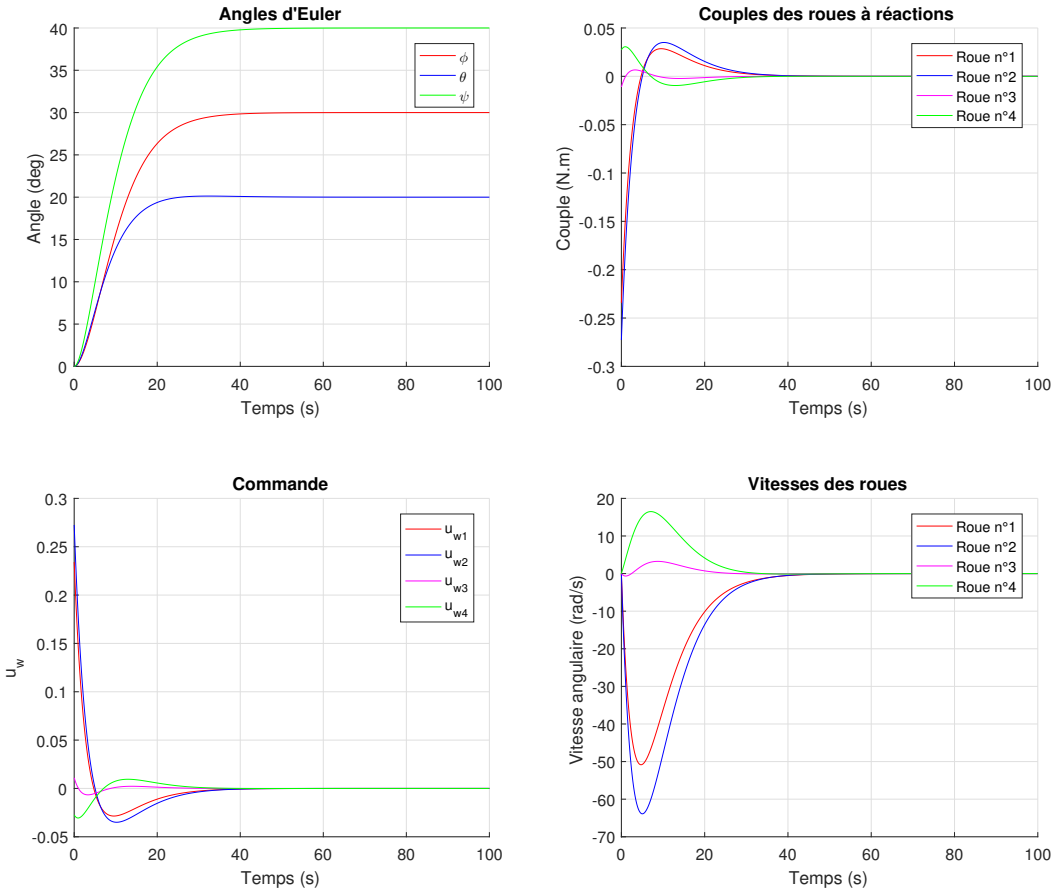

Figure 5.3 Temps de réponse pour l'optimisation de l'angle d'inclinaison

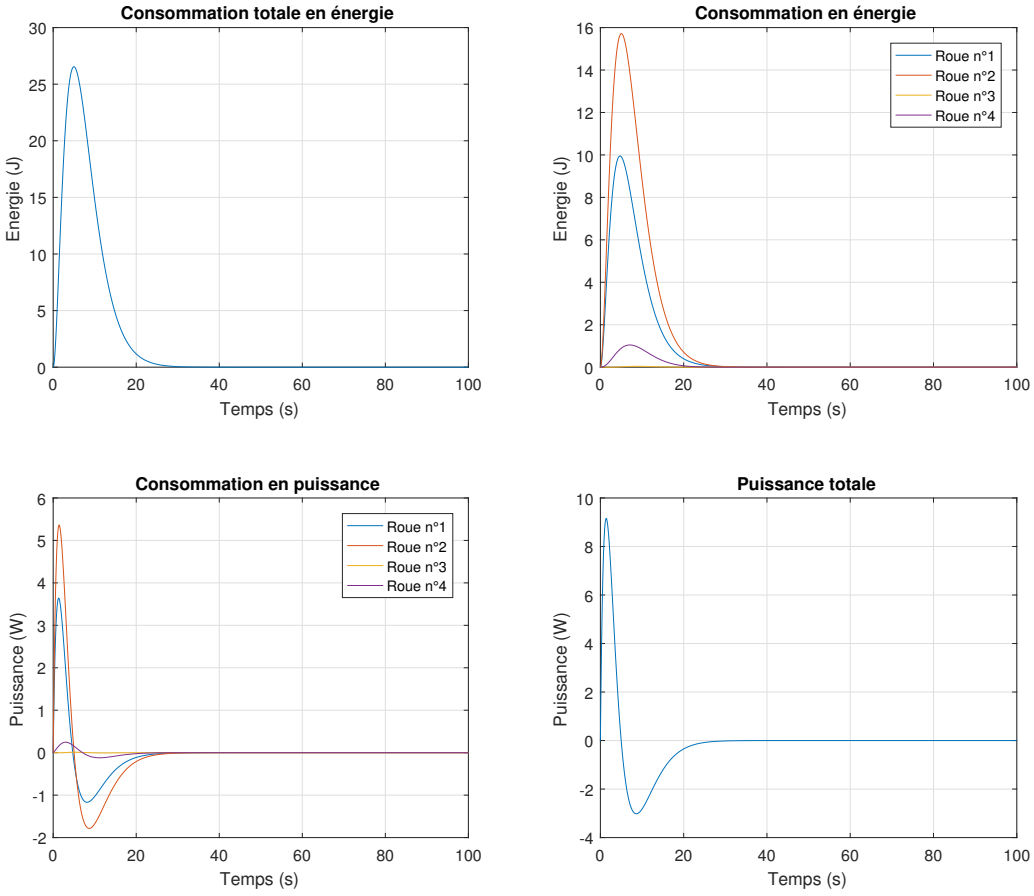

Figure 5.4 Énergie et puissance consommées

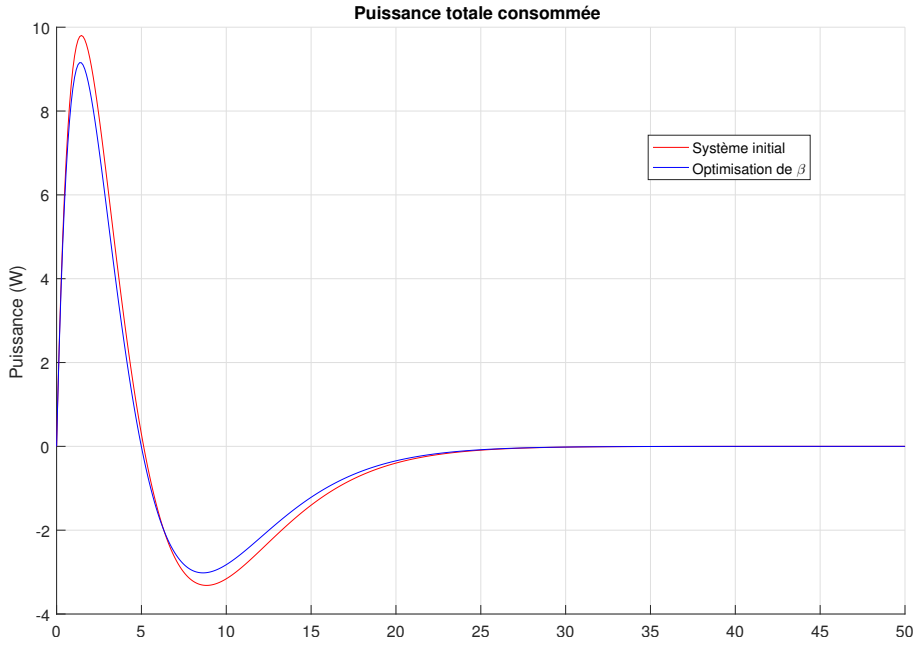

Figure 5.5 Puissance consommée avec et sans optimisation de l'angle β

Optimisation de β et des gains du correcteur ($\mathbf{K}_p$ et $\mathbf{K}_d$)

Dans cette partie, nous définissons l'angle d'inclinaison et les gains du correcteur en tant que paramètres variables. Le grand nombre de combinaison possible des paramètres a pour effet de ne pas mener `systune` au même résultat à chaque exécution. C'est à cet instant qu'il est nécessaire de faire intervenir la fonctionnalité `RandomStarts` : plus le nombre qui y est associé est grand, plus le résultat renvoyé par `systune` sera proche de la combinaison optimale des paramètres. Après plusieurs essais, les valeurs suivantes ont été retenues pour les paramètres :

$$\begin{aligned} \beta &= 27.35^\circ \\ \mathbf{K}_p &= \begin{bmatrix} 0.9007 & 0 & 0 \\ 0 & 1.0451 & 0 \\ 0 & 0 & 0.7472 \end{bmatrix} \\ \mathbf{K}_d &= \begin{bmatrix} 4.6060 & 0 & 0 \\ 0 & 5.5289 & 0 \\ 0 & 0 & 3.8585 \end{bmatrix} \end{aligned} \tag{5.5}$$

On peut remarquer que la valeur optimale de l'angle d'inclinaison a changé, ce qui montre bien

que les paramètres ne sont pas indépendants. Cela montre également qu'il est bien nécessaire d'optimiser en parallèle tous les paramètres, et non séparément, lorsque l'on effectue le co-design d'un système. Les nouvelles réponses temporelles obtenues sont données dans les figures 5.6 et 5.7.

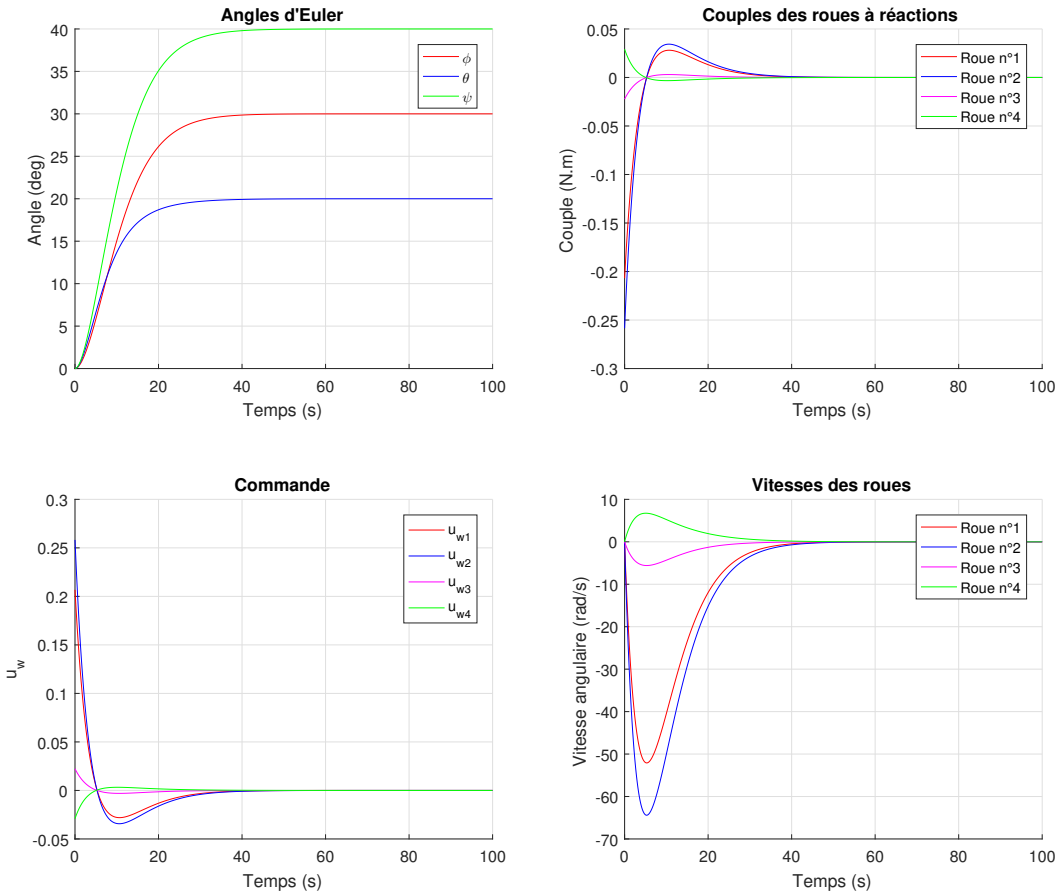

Figure 5.6 Réponses temporelles pour l'optimisation de β, $\mathbf{K}_p$ et $\mathbf{K}_d$

Dans ce second cas, la condition sur le dépassement en réponse à l'échelon est bien respectée grâce à l'optimisation des gains du correcteur.

Comparaison des consommations de puissance

Nous pouvons une fois de plus comparer les consommations de puissance entre les deux optimisations réalisées précédemment afin de confirmer l'intérêt de l'optimisation de plusieurs paramètres en simultané grâce à `systune` .

Une fois encore la figure 5.8 confirme que nous sommes parvenus à réduire la puissance consommée tout en respectant les contraintes de performance imposées. La consommation

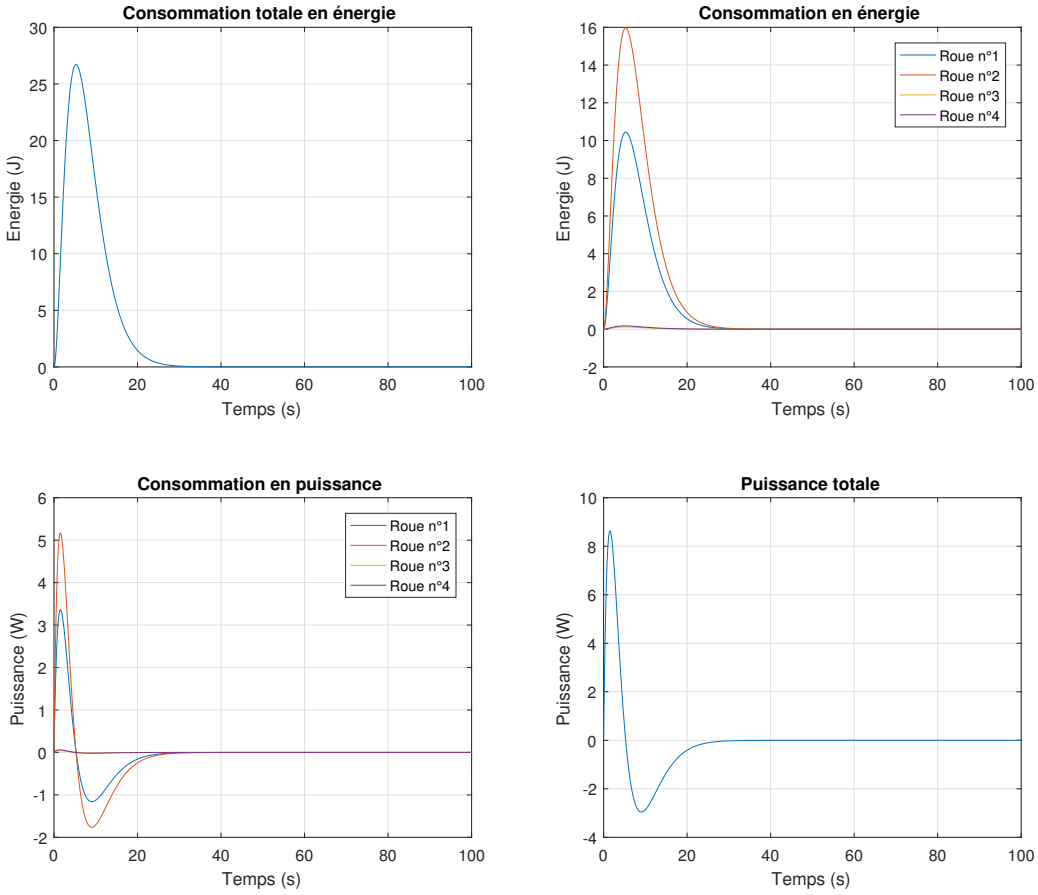

Figure 5.7 Énergie et puissance consommées

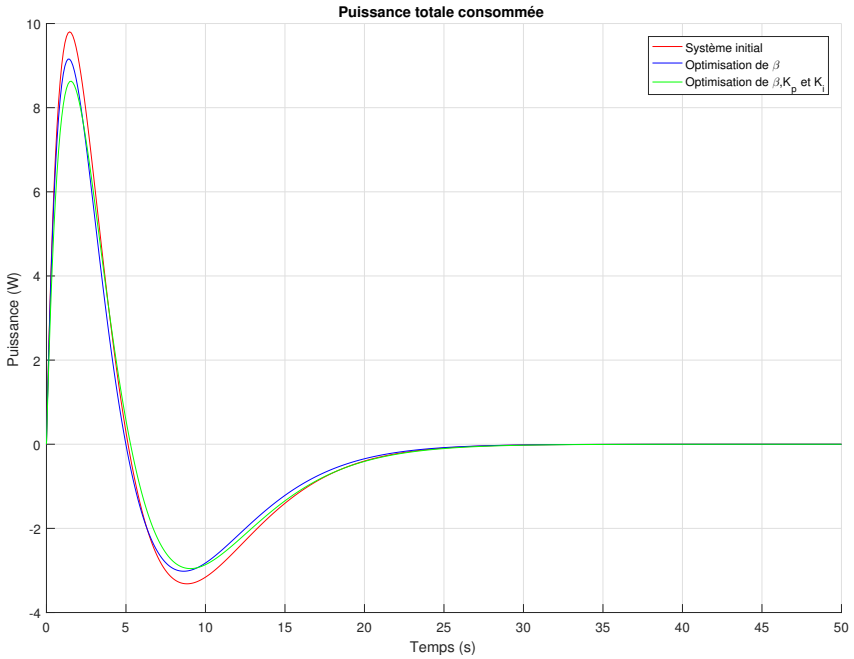

Figure 5.8 Consommation de puissance du système initial, avec β optimisé et β, $\mathbf{K}_p$ et $\mathbf{K}_d$ optimisés

de puissance est par ailleurs plus faible que lors de la précédente optimisation.

Optimisations simultanées de β, $\mathbf{K}_p$, $\mathbf{K}_d$ et de l'inertie J_w

Enfin, il s'agit d'optimiser ici tous les paramètres variants que nous avons définis en même temps en ajoutant la valeur de l'inertie des roues de réaction. Comme dans le cas précédent nous allons utiliser la valeur de 100 pour `RandomStarts` ; une valeur plus grande bien que garantissant une meilleure probabilité de trouver une combinaison optimale de valeurs s'est avérée rallonger le temps de calcul, sans pour autant fournir de résultats plus concluants. Les valeurs alors obtenues sont les suivantes :

$$\begin{aligned} \beta &= 27.28° \\ \mathbf{K}_p &= \begin{bmatrix} 0.9054 & 0 & 0 \\ 0 & 1.0645 & 0 \\ 0 & 0 & 0.7509 \end{bmatrix} \\ \mathbf{K}_d &= \begin{bmatrix} 4.6483 & 0 & 0 \\ 0 & 5.6047 & 0 \\ 0 & 0 & 3.8719 \end{bmatrix} \\ J_w &= 92.10^{-4} \text{ kg.m}^2 \end{aligned} \tag{5.6}$$

Il est important de remarquer ici que la valeur de l'angle d'inclinaison a une nouvelle fois changé ce qui est dû à l'interdépendance des paramètres entre eux. Le co-design permet donc d'optimiser plusieurs plusieurs paramètres à la fois, là où l'on se contentait auparavant d'optimiser les paramètres les uns après les autres dans la mesure du possible. Les réponses temporelles pour ces nouvelles réponses optimisées sont présentées dans les figures 5.9 et 5.10.

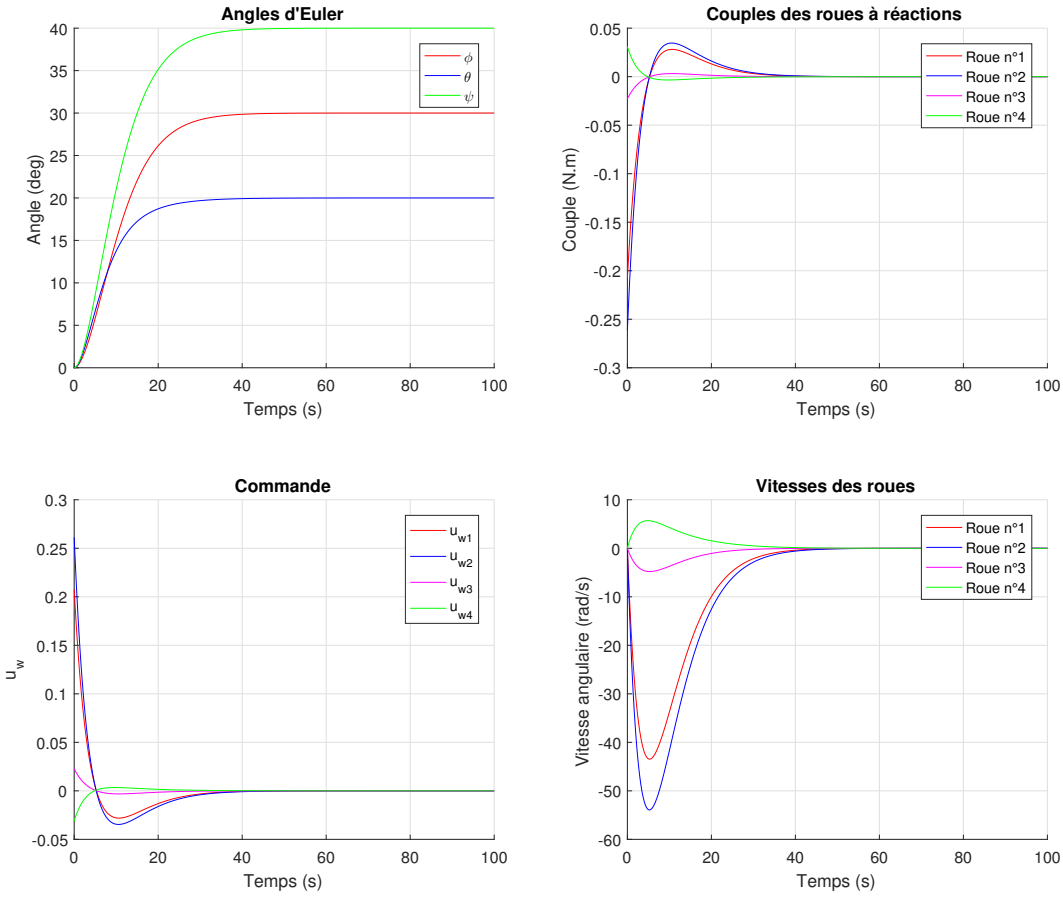

Figure 5.9 Réponses temporelles pour l'optimisation de tous les paramètres

Comparaison des étapes d'optimisation

Afin de pouvoir conclure quant à l'intérêt de la méthode présentée, nous pouvons comparer les trois optimisations précédentes et le système que nous avions initialement. Les résultats sont donnés en figure 5.11. Il apparaît clairement que le nombre de paramètres réglables joue un rôle important dans le co-design. Bien que la différence ne soit pas très importante, il est important de remarquer que le cas initial était d'ores et déjà optimisé pour les performances souhaitées. Il est donc possible de conclure que l'utilisation de `systune` a bel et bien permis d'obtenir des performances encore supérieures grâce au co-design.

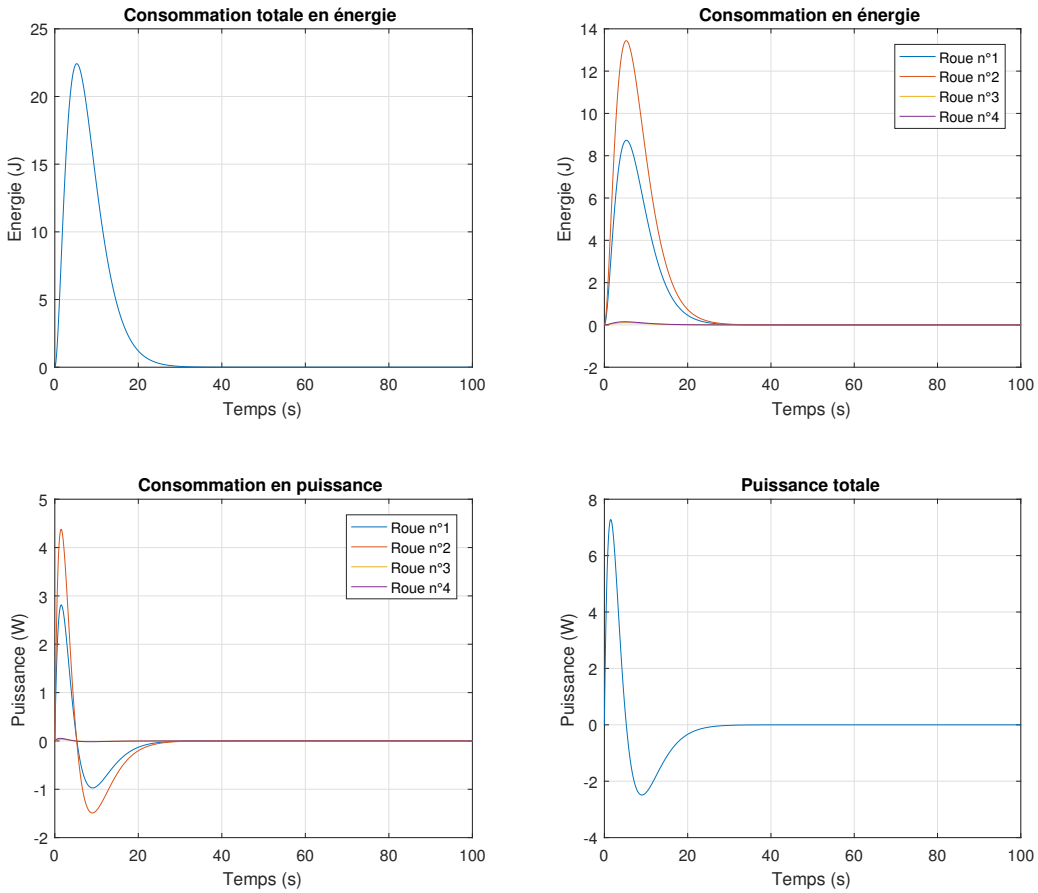

Figure 5.10 Consommation d'énergie et de puissance pour l'optimisation de tous les paramètres

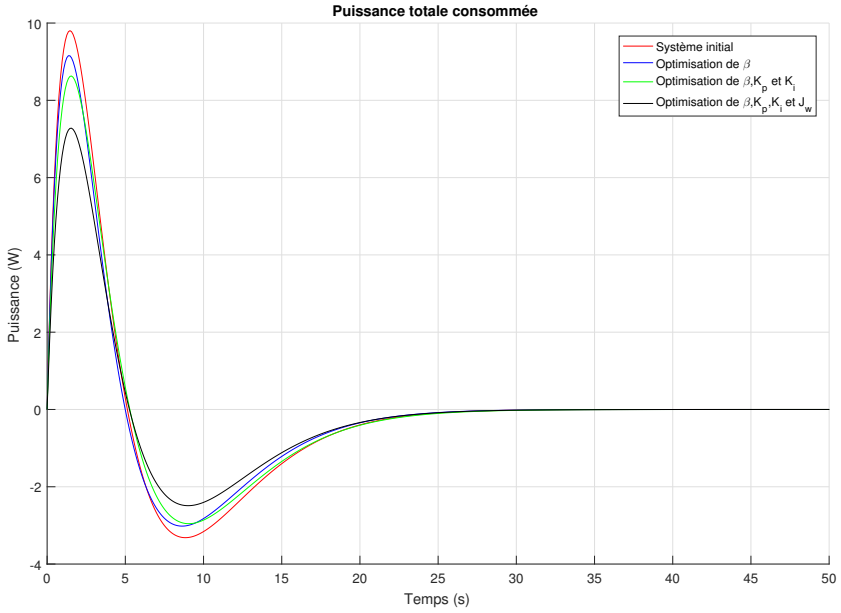

Figure 5.11 Comparaison de la puissance consommée entre le système initial et les trois niveaux d'optimisations

5.3.3 Valeur de la norme

Afin de conclure ce chapitre, nous pouvons comparer la valeur des coefficients γ entre les différentes optimisations réalisées. Ces valeurs représentent la valeur minimale de la norme $\mathcal{H}_\infty$ et donc de la conformité du système en boucle fermée aux performances que nous lui avons imposées. Ainsi, plus celles-ci sont proches de 1, plus les contraintes sont satisfaites.

Par ailleurs, nous avons défini deux types de contraintes de conception les `HardGoals` et les `SoftGoals` ; il y a donc deux coefficients par simulation. Les `HardGoals` doivent être au plus proche de 1, tandis que les seconds peuvent être plus élevés, car les `SoftGoals` ont ici pour but de diminuer la consommation de puissance au maximum tant que les premiers sont respectés.

Tableau 5.3 Valeurs de γ selon les cas d'optimisation

Optimisation	β	β, K_p et K_d	β, K_p, K_d et J_w
HardGoals	2.9865	1.3694	1.1459
SoftGoals	56.3	53.9	44.3

Comme on peut le voir la valeur de γ diminue avec le nombre de paramètres variants ce qui est conforme à l'intuition, car l'algorithme a alors plus de liberté pour améliorer les performances du système.

CHAPITRE 6 OPTIMISATION DU SYSTÈME D'ORIENTATION D'UN CUBESAT

Après avoir conclu que `systune` permettait bel et bien d'optimiser un système mécanique et son correcteur, nous allons appliquer cette méthode sur un satellite en particulier. Le système qui sera étudié sera de type Cubesat, car les résultats développés dans ce mémoire sont utilisés en parallèle pour développer le nanosatellite *ORUS* de la société étudiante PolyOrbite.
L'étude sera donc portée sur un Cubesat 3 Unités (Fig. 6.1) dont le système d'orientation est constitué de 4 roues de réaction et de 3 magnéto-coupleurs. Ce choix d'actionneurs est courant pour ce type de satellite, car il permet une grande précision de pointage et garantit la sécurité du système en cas de panne d'un des composants.

Figure 6.1 Cubesat 3U (source : n-avionics.com)

La matrice d'inertie **J** d'un tel satellite est estimée à :

$$\mathbf{J} = \begin{bmatrix} J_X & 0 & 0 \\ 0 & J_Y & 0 \\ 0 & 0 & J_Z \end{bmatrix} = \begin{bmatrix} 0.026 & 0 & 0 \\ 0 & 0.026 & 0 \\ 0 & 0 & 0.004 \end{bmatrix} \text{ kg.m}^2 \tag{6.1}$$

Notre but est d'obtenir un système respectant les performances détaillées plus bas, sous la

contrainte de réduire la masse et la consommation d'énergie du satellite. Nous allons donc présenter dans ce chapitre les différentes étapes de paramétrisation et d'implantation sous MATLAB pour arriver au résultat escompté pour deux modes de fonctionnement du satellite. Dans un premier temps, nous considérons le mode de pointage à l'aide des roues de réaction puis le mode désaturation des roues à l'aide de magnéto-coupleurs est abordé.

6.1 Optimisation du système de pointage du Cubesat

Cette première section est dédiée au système de pointage du satellite, composé de l'association des 4 roues de réaction dans une structure pyramidale telle que décrite dans le chapitre précédent. Cependant, il est à présent question d'optimiser directement la géométrie des roues et leur disposition afin de se placer dans un exemple concret de ce que cette méthode peut apporter au co-design. Afin de prouver que la démarche développée dans ce mémoire peut être appliquée à différentes modélisations, nous l'appliquerons à deux géométries de roue différentes : des roues tubulaires (en forme de cylindre «creux»), et des roues cylindriques «pleines».

6.1.1 Définition et paramétrisation du Cubesat

Linéarisation du satellite

Comme précédemment, grâce à la fonction `linmod` de MATLAB on linéarise le satellite autour du point d'équilibre $[0\,0\,0]^\top$. Le modèle linéarisé est ce qui nous permettra d'utiliser `systune` pour optimiser le système, avant d'implanter les valeurs obtenues dans le système non linéaire pour en vérifier les performances. On obtient ainsi un système LTI qui modélise les dynamiques du cubesat. La linéarisation donne les quatre matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ et $\mathbf{D}$ ci-dessous :

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.2}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 38.4615 & 0 & 0 & 0 & 0 & 0 \\ 0 & 38.4615 & 0 & 0 & 0 & 0 \\ 0 & 0 & 250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.3}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.4}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 38.4615 & 0 & 0 & 0 & 0 & 0 \\ 0 & 38.4615 & 0 & 0 & 0 & 0 \\ 0 & 0 & 250 & 0 & 0 & 0 \end{bmatrix} \tag{6.5}$$

Matrice de distribution

Les roues sont présentées en disposition pyramidale comme cela a été présenté au Chapitre 5. Afin de pouvoir utiliser `systune` on utilise la paramétrisation décrite dans l'équation 5.3 à l'aide des paramètres a et b.

Géométrie de la roue creuse

La roue d'inertie étudiée ici est axisymétrique, ses paramètres géométriques sont définis sur la figure 6.2. L'inertie J_w et sa masse m sont données par les formules suivantes [29] :

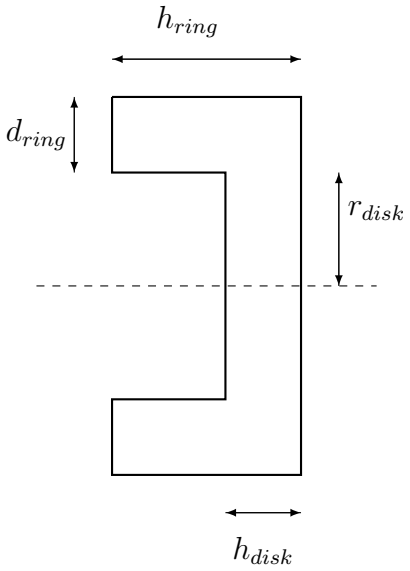

Figure 6.2 Vue en coupe de la roue d'inertie creuse

$$J_w = \rho\frac{\pi}{2}(h_{ring}(r_{row}^4 - r_{disk}^4) + h_{disk}r_{disk}^4)$$

$$m = \rho\pi(r_{row}^2 h_{ring} - r_{disk}^2(h_{ring} - h_{disk}))$$

$$\text{avec} \quad r_{row} = r_{disk} + d_{ring}$$

On utilise comme matériau du fer dont la masse volumique vaut $\rho = 7.87 \times 10^3\,\text{kg.m}^3$. Ce matériau a été choisi pour ses propriétés de dégazage dans l'espace et par comparaison de son coût avec d'autres matériaux. Ce sont ces paramètres géométriques que nous cherchons à optimiser en plus des gains du correcteur. On utilise pour valeurs de départ celle de la conception préliminaire fournie par l'équipe de PolyOrbite (Tab. 6.1).

Tableau 6.1 Dimensions géométriques préliminaires de la roue n°1

Paramètre	Valeur	Unité
d_{ring}	4	mm
h_{ring}	11.8	mm
h_{disk}	4.1	mm
r_{disk}	10	mm
m	$38.1.10^{-3}$	kg
J_w	$4.652.10^{-6}$	kg.m^2

On définit alors ces grandeurs en tant que paramètres `realp` :

```
dr_m = 4/1000;
hr_m = 11.8/1000;
hd_m = 4.1/1000;
rd_m = 10/1000;

r_disk = realp('r_disk',rd_m);
d_ring = realp('d_ring',dr_m);
h_disk = realp('h_disk',hd_m);
h_ring = realp('h_ring',hr_m);

r_row = r_disk+d_ring;
m     = rho*pi*r_row^2*h_ring-rho*pi*r_disk^2*(h_ring-h_disk);
Jw    = rho*(pi/2)*(h_ring*(r_row^4-r_disk^4)+h_disk*r_disk^4);
```

Afin de limiter les intervalles de variations des paramètres définis ci-dessus, on définit également des valeurs minimales et maximales pour chacun d'entre eux.

```
v = 25/100;
r_disk_min = rd_m*(1-v);
r_disk_max = rd_m*(1+v);

d_ring_min = 0;
d_ring_max = dr_m*(1+v);

h_disk_min = hd_m*(1-v);
h_disk_max = h_ring_max;

h_ring_min = h_disk_min;
h_ring_max = hr_m*(1+v);
```

On autorise ainsi les paramètres à varier de $\pm 25\%$ de leur valeur initiale, sauf dans 3 cas particuliers où l'on modifie les bornes :

- pour d_{ring}, on prend pour valeur minimale 0 pour inclure le cas d'une roue cylindrique ;
- pour h_{disk}, cette grandeur ne peut pas être plus grande que celle de h_{ring} ;
- pour h_{ring}, cette grandeur ne peut pas être plus petite que celle de h_{disk}.

Géométrie de la roue cylindrique

On considère ici une deuxième géométrie de roue axisymétrique sur laquelle on souhaite faire du co-design. Cette géométrie et ses paramètres sont représentés sur la figure 6.3. La démarche pour effectuer le co-design est la même que précédemment.

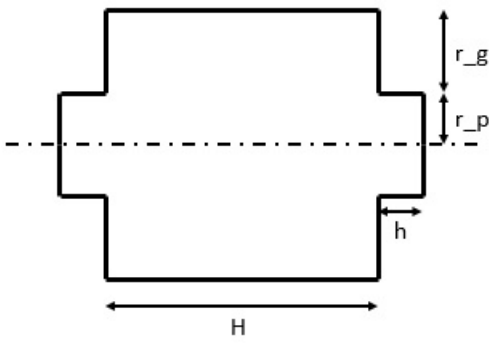

Figure 6.3 Vue en coupe de la roue d'inertie n°2

L'inertie J_w et sa masse m sont données par les formules suivantes :

$$J_w = \rho\frac{\pi}{2}(HR^4 + 2hr_p^4)$$

$$m = \rho\pi(HR^2 + 2hr_p^2)$$

$$\text{avec} \quad R = r_p + r_g$$

On prend pour valeurs initiales celles du tableau 6.2.

Tableau 6.2 Dimensions géométriques préliminaires de la roue n°2

Paramètre	Valeur	Unité
r_p	2.5	mm
r_g	11.5	mm
h	2	mm
H	8	mm
m	$39.4.10^{-3}$	kg
J_w	$3.8702.10^{-6}$	$kg.m^2$

De même que précédemment, on définit ces paramètres par des fonctions `realp` :

```
rp_m = 2.5/1000;
rg_m = 11.5/1000;
h_m  = 2/1000;
H_m  = 8/1000;

rp = realp('rp',rp_m);
rg = realp('rg',rg_m);
H  = realp('H',H_m);
h  = realp('h',h_m);

R  = rp+rg;
m  = rho*pi*(R^2*H+2*rp^2*h);
Jw = rho*(pi/2)*(H*R^4+2*h*rp^4);
```

Ainsi que les intervalles de définition :

```
v=25/100;

rp_min = 0;
rp_max = rp_m*(1+v);
```

```
rg_min = rg_m*(1-v);
rg_max = rg_m*(1+v);

H_min = H_m*(1-v);
H_max = H_m*(1+v);

h_min = 0;
h_max = h_m*(1+v);
```

On autorise r_p et h à prendre la valeur nulle pour inclure le cas d'un cylindre parfait.

Architecture du correcteur

Comme indiqué dans le chapitre précédent, on choisit d'utiliser un correcteur de type Proportionnel Dérivé (PD). Un choix classique de valeurs pour les gains du correcteur est donné par la formule suivante [32] :

$$\begin{aligned} \mathbf{K}_p &= \omega_n^2 \mathbf{J} \\ \mathbf{K}_d &= 2\zeta\omega_n \mathbf{J} \end{aligned} \tag{6.6}$$

où ω_n et ζ sont respectivement la pulsation propre et l'amortissement de la réponse temporelle souhaitée.

Dans notre cas, si l'on cherche à obtenir un temps de réponse à 2% T_r de 25 s ainsi qu'un dépassement relatif maximal de 5% on obtient les deux coefficients précédents par les formules :

$$\begin{aligned} \zeta &= 0.7 \\ \omega_n &= \frac{4}{\zeta T_r} \end{aligned} \tag{6.7}$$

Ceci nous donne pour gains du correcteur :

$$\mathbf{K}_{p,i} = \begin{bmatrix} 0.0014 & 0 & 0 \\ 0 & 0.0014 & 0 \\ 0 & 0 & 0.0002 \end{bmatrix} \tag{6.8}$$
$$\mathbf{K}_{d,i} = \begin{bmatrix} 0.0083 & 0 & 0 \\ 0 & 0.0083 & 0 \\ 0 & 0 & 0.0013 \end{bmatrix}$$

On peut vérifier que pour ces valeurs, le satellite respecte bien les contraintes en réponse temporelles pour une attitude désirée de $[30\,20\,-40]^\top$. Les réponses sont tracées sur la figure 6.4.

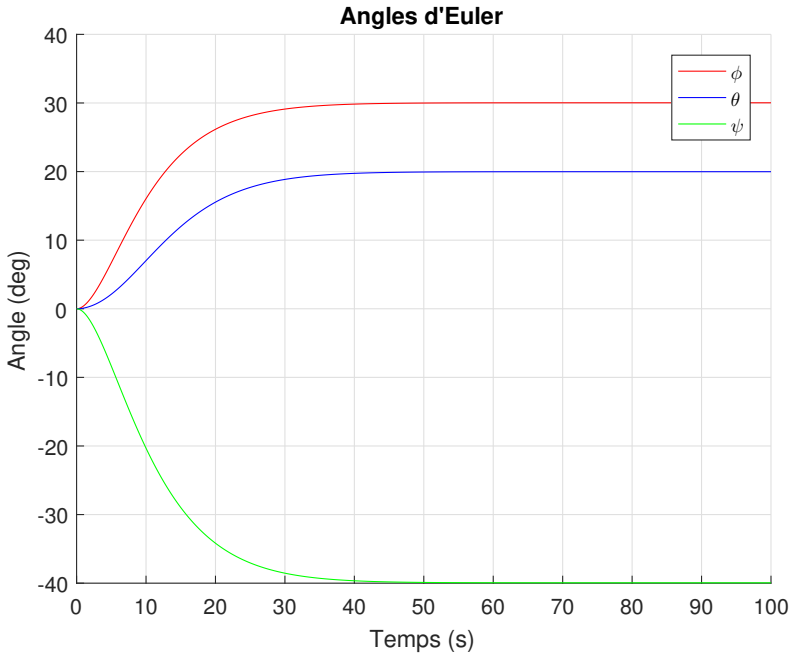

Figure 6.4 Commande en attitude du satellite initial

On définit les gains du correcteur de la même façon que celle appliquée dans le chapitre précédent. Cependant les valeurs initiales de ceux-ci sont fixées grâce au calcul ci-dessus, afin de permettre à `systune` de converger plus rapidement vers une solution fonctionnelle et surtout d'éviter que l'algorithme ne parvienne pas à stabiliser le système.

6.1.2 Contraintes de conception

Cette partie est dédiée à la définition des contraintes qui seront imposées à `systune` afin que le système final respecte le cahier des charges tout en optimisant le système d'actionneur.

Réponse temporelle

Comme nous l'avons vu dans le chapitre précédent, la fonction `StepTracking` permet d'imposer la réponse temporelle du système à partir du temps de réponse et de l'amortissement souhaités. Avec les performances qui ont été décrites plus haut, on peut définir la contrainte en temps de réponse `TrackReq`. Afin de s'assurer que cette contrainte soit assurée, elle doit être définie en tant que `HardGoal`.

```
overshoot     = 5;
zeta          = 0.7;
Tr            = 25; % temps de reponse a 2% desire
tau_response = zeta*Tr/4;

TrackReq = TuningGoal.StepTracking({'ref_phi','ref_theta','ref_psi
    '},{'phi','theta','psi'},tau_response,overshoot);
TrackReq.RelGap = 0.1;
```

Minimisation de la consommation d'énergie

Afin de limiter également la puissance fournie par les moteurs entraînant les roues de réactions ainsi que de limiter l'accélération angulaire des roues, nous définissons une seconde contrainte. Cette contrainte, que l'on nommera `CommandReq`, doit être imposée entre l'attitude commandée et l'accélération angulaire des roues comme montrée ci-dessous :

```
max_speed   = 300;
CommandReq = TuningGoal.Gain({'ref_phi','ref_theta','ref_psi'},{'
    Omega_d1','Omega_d2','Omega_d3','Omega_d4'},max_speed);
```

La valeur de gain maximal a été obtenue après plusieurs essais comme étant la mieux adaptée afin de ne pas trop contraindre `systune` et de l'empêcher de respecter les autres contraintes.

De plus, cette contrainte est définie en tant que, `SoftGoal` car nous cherchons à minimiser la puissance nécessaire au mouvement sous couvert que les contraintes en temps de réponse et minimisation de masse soient respectées. Toutefois, afin d'étudier l'impact de ce choix,

`systune` est exécuté en changeant la contrainte en `HarGoal` plus tard dans cette section afin de comparer les résultats obtenus par les deux optimisations.

Réduction de la masse

On souhaite réduire la masse embarquée associée aux roues de réaction en optimisant la forme de celles-ci. Pour cela, nous cherchons à imposer à chaque roue une masse de 25 g afin de réduire la masse totale du satellite. On crée donc une fonction de transfert m_{TF} de valeur m :

$$\frac{m_y}{m_u} = m_{TF} = m \tag{6.9}$$

On impose ensuite une valeur de gain maximal à cette fonction de transfert correspondant à la masse que l'on veut donner à la roue m_{max} :

```
m_max = 0.025; % pour 25 g
m_Req = TuningGoal.Gain({'m_u'},{'m_y'},m_max);
```

Le but de ces travaux étant de réduire la masse tout en obtenant les meilleures performances possibles, cette contrainte sera imposée en tant que `HardGoal`.

Augmentation de l'inertie

Par la contrainte définie ci-dessus, l'algorithme va donc optimiser les paramètres afin de la réduire en deçà de la limite imposée (ici de 25 g). Cependant l'expérience a permis de conclure qu'il est nécessaire de poser une contrainte sur l'inertie de la roue également.

En effet, si l'on n'impose pas à `systune` de maximiser en parallèle l'inertie J_w des roues de réactions la forme finale de la roue ne sera pas adaptée aux contraintes de limitation en consommation de puissance. Il convient donc de définir une fonction de transfert $J_{w_{TF}}$ d'expression :

$$\frac{J_{w_y}}{J_{w_u}} = J_{w_{TF}} = \frac{1}{J_w} \tag{6.10}$$

On impose ensuite un requis $J_{w_{Req}}$ sur cette fonction en tant que `SoftGoal`, de façon à s'assurer que la forme de la roue minimise la masse tout en augmentant au possible l'inertie de la roue.

```
Jw_tf   = tf(1/Jw);
Jw_tf.u = 'Jw_tf_u';
Jw_tf.y = 'Jw_tf_y';
```

```
Jw_Req = TuningGoal.Gain({'Jw_tf_u'},{'Jw_tf_y'},1/Jwi);
```

6.1.3 Résultats pour la roue creuse

Géométrie optimisée

Les valeurs obtenues après optimisation pour les grandeurs géométriques de la roue sont décrites dans le tableau 6.3.

Tableau 6.3 Dimensions géométriques de la roue creuse obtenues par co-design

Paramètre	Valeur	Unité
d_{ring}	5	mm
h_{ring}	3.54	mm
h_{disk}	3.08	mm
r_{disk}	12.5	mm
m	25	g
J_w	4.0308×10^{-6}	kg.m^2

La géométrie de la roue a bien été optimisée de sorte à obtenir une masse de 25 g tout en gardant une inertie comparable à sa valeur initiale ($J_{w,i} = 4.652 \times 10^{-6}$ kg.m^2). De même nous obtenons de nouvelles valeurs pour les angles définissant la disposition pyramidale des roues :

Tableau 6.4 Angles de la structure pyramidale avec roues creuses

Paramètre	Valeur	Unité
a	0.987	
b	0.1385	
α	89.9	°
β	15.77	°

Simulation de la réponse temporelle

Les gains du correcteur retournés par `systune` sont :

$$\mathbf{K}_p = \begin{bmatrix} 0.0012 & 0 & 0 \\ 0 & 0.0011 & 0 \\ 0 & 0 & 0.0002 \end{bmatrix} \\ \mathbf{K}_d = \begin{bmatrix} 0.0083 & 0 & 0 \\ 0 & 0.0080 & 0 \\ 0 & 0 & 0.0013 \end{bmatrix} \tag{6.11}$$

On peut noter que ceux-ci sont proches des valeurs données comme valeurs initiales. Cela est cohérent, car nous avions choisi comme initialisation des valeurs permettant déjà d'obtenir les performances souhaitées, dans le but de guider l'algorithme vers des valeurs stabilisant le système dès les premières itérations. Nous pouvons à présent tracer les réponses temporelles du système lorsqu'on lui commande une attitude à atteindre, par exemple $[30° \; 20° \; -40°]^\top$ (Fig. 6.5, 6.6, 6.7 et 6.8). La réponse du système en pointage est satisfaisante et le système converge bien vers l'orientation commandée en environ 25 s.

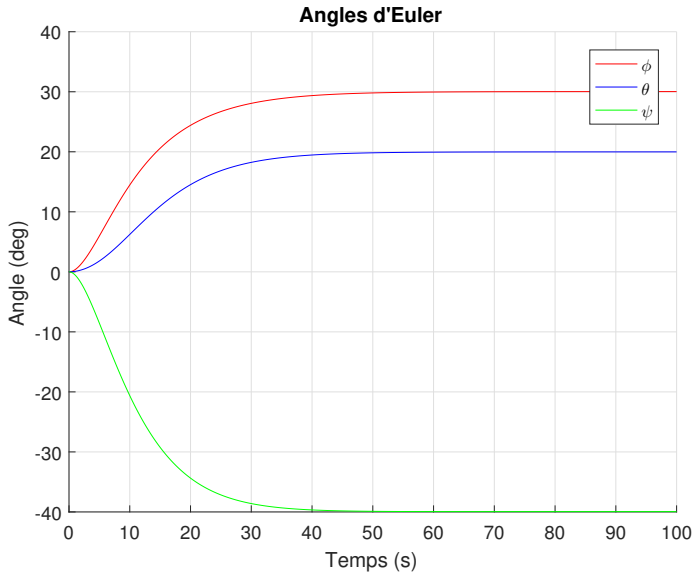

Figure 6.5 Réponses temporelles - Roue Creuse

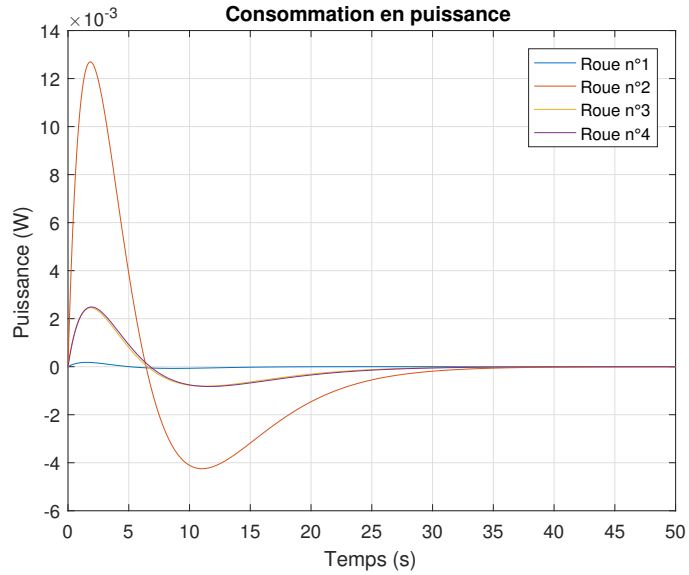

Figure 6.6 Consommation en puissance des 4 roues - Roue Creuse

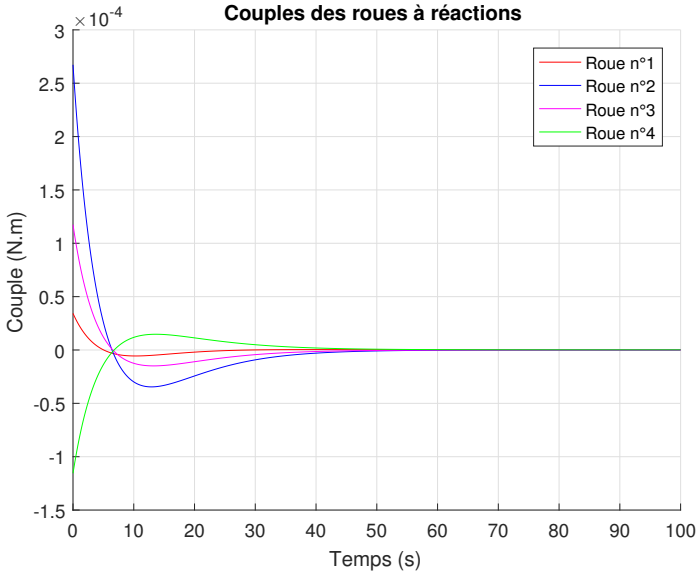

Figure 6.7 Couples des roues - Roue Creuse

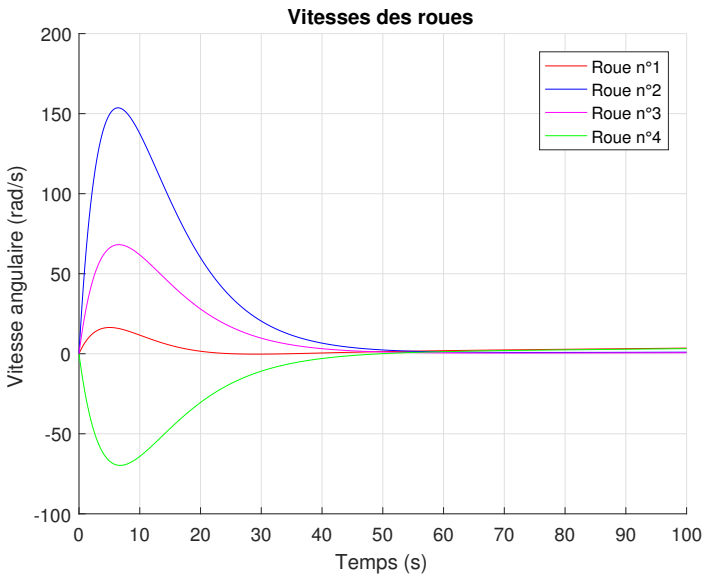

Figure 6.8 Vitesses de rotation des roues - Roue Creuse

On compare enfin sur la figure 6.9 la consommation nécessaire au mouvement commandé entre le système initial et le système obtenu grâce à `systune`. Il apparaît clairement qu'en plus de fournir une géométrie optimisée, la consommation en puissance a également été grandement réduite. Dans le cas de notre simulation, on obtient un pic de consommation presque 2 fois inférieur.

Ceci nous permet de confirmer l'intérêt de la méthode développée dans ce mémoire : `systune` a permis d'obtenir un système plus performant avec des actionneurs de moindre masse et dont le design a été optimisé en parallèle du correcteur.

Respect des contraintes de conception

La fonction `viewSpec` nous permet ensuite de vérifier que les contraintes imposées sont bien respectées une à une. Seule la condition du gain pour l'inertie n'est pas représentée, car cette contrainte n'est pas rencontrée. Nous ne présenterons pas ces graphiques pour les optimisations réalisées par la suite, mais la fonction `viewSpec` aurait permis de les obtenir de la même façon. Nous pouvons voir ici que les réponses sont très semblables à celles souhaites, malgré un dépassement légèrement plus grand pour les commandes en ϕ et θ (Figs. 6.10, 6.11 et 6.12)

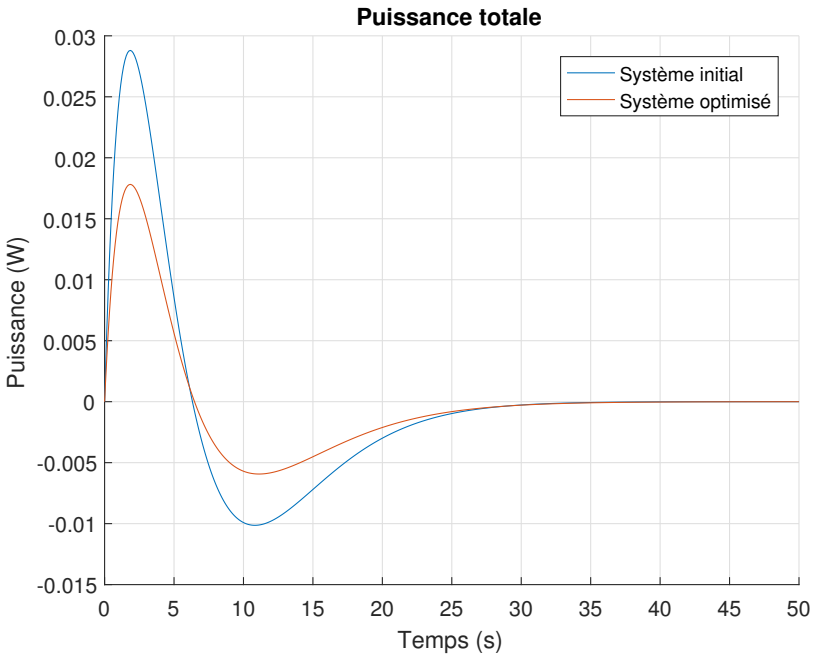

Figure 6.9 Comparaison de la consommation de puissance entre le système initial et le système optimisé

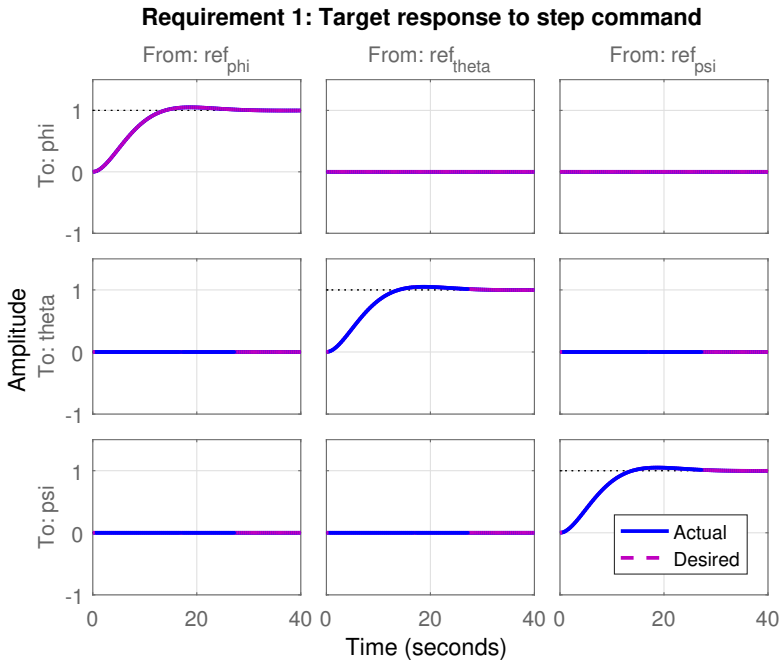

Figure 6.10 Contrainte en temps de réponse à un échelon pour ϕ, θ et ψ

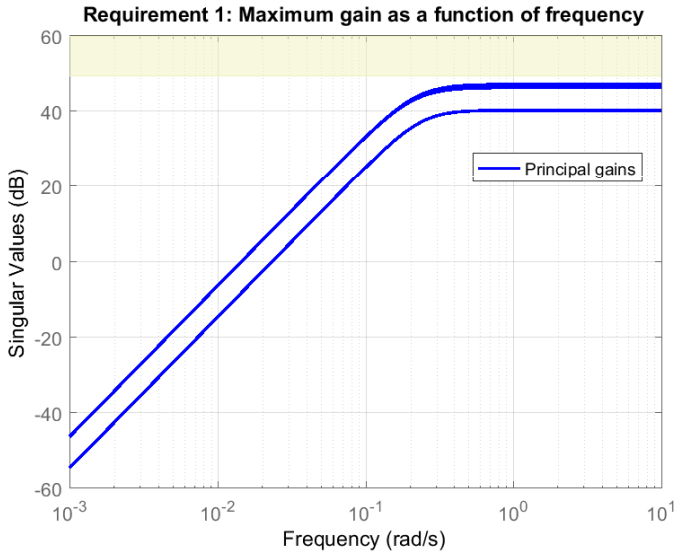

Figure 6.11 Contraintes en amplitude de commande

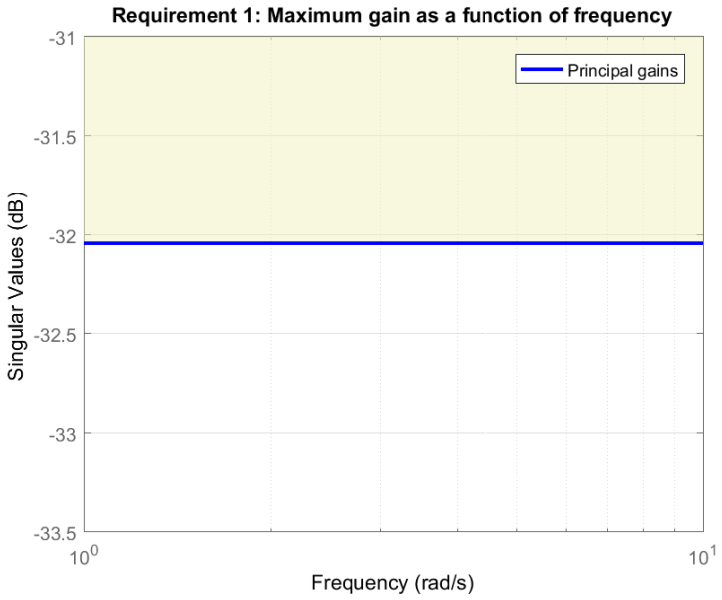

Figure 6.12 Contrainte sur la masse des roues

6.1.4 Résultats pour la roue cylindrique

Nous appliquons ici les mêmes contraintes à la seconde morphologie de la roue. Les résultats ci-dessous permettent de confirmer que `systune` obtient bien une roue avec la masse souhaitée, cependant l'algorithme ne permet plus de limiter autant la consommation en énergie.

Certaines valeurs étant très proches de 0, la liberté a été prise d'arrondir (Tab. 6.7).

Tableau 6.5 Dimensions géométriques de la roue n°2 obtenues par co-design

Paramètre	Valeur	Unité
r_p	0	mm
r_g	13	mm
h	0	mm
H	6	mm
m	25	g
J_w	2.175×10^{-6}	kg.m^2

On obtient également des valeurs pour les angles de la structure pyramidale soutenant les roues (Fig. 6.8)

Tableau 6.6 Angles de la structure pyramidale avec roues cylindriques

Paramètre	Valeur	Unité
a	0.1	
b	0.23	
α	11.42	°
β	25.9	°

Simulations

Les gains retournés par `systune` sont :

$$\mathbf{K}_p = \begin{bmatrix} 0.0012 & 0 & 0 \\ 0 & 0.0010 & 0 \\ 0 & 0 & 0.0002 \end{bmatrix}$$
$$\mathbf{K}_d = \begin{bmatrix} 0.0074 & 0 & 0 \\ 0 & 0.0064 & 0 \\ 0 & 0 & 0.0013 \end{bmatrix} \tag{6.12}$$

Les gains du correcteur demeurent proches de ceux obtenus pour la roue précédente. On trace à nouveau les réponses temporelles pour la deuxième géométrie de roue pour la même commande (Figs. 6.13, 6.14, 6.15 et 6.16). Une nouvelle fois les temps de réponse sont bien respectés malgré le changement de géométrie.

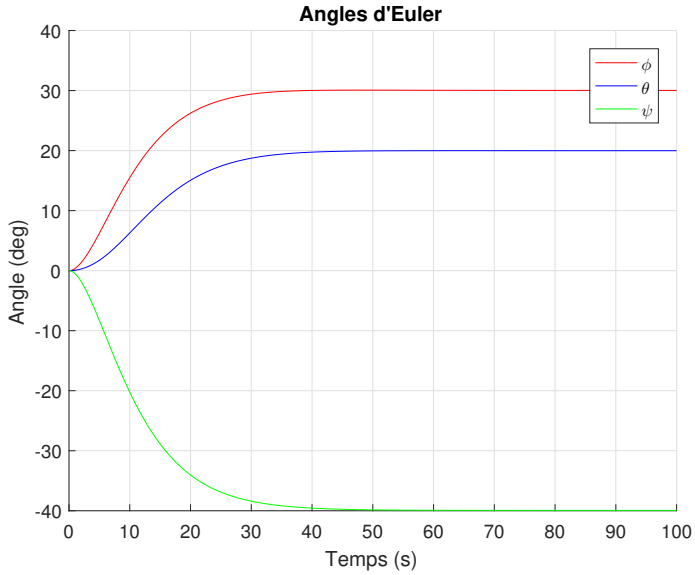

Figure 6.13 Réponses temporelles - Roue Cylindrique

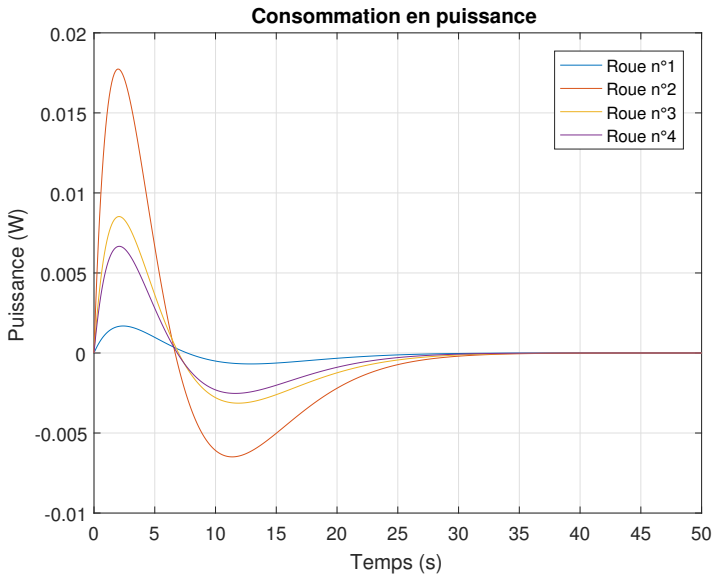

Figure 6.14 Consommation en puissance des 4 roues - Roue Cylindrique

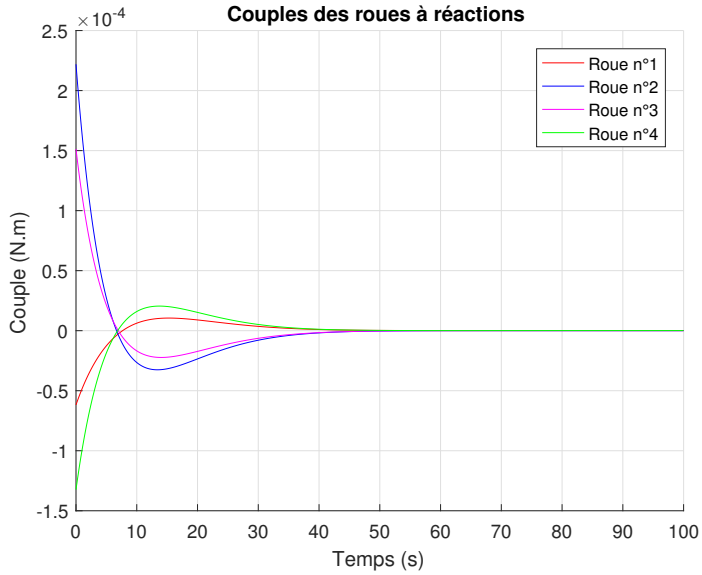

Figure 6.15 Couples des roues - Roue Cylindrique

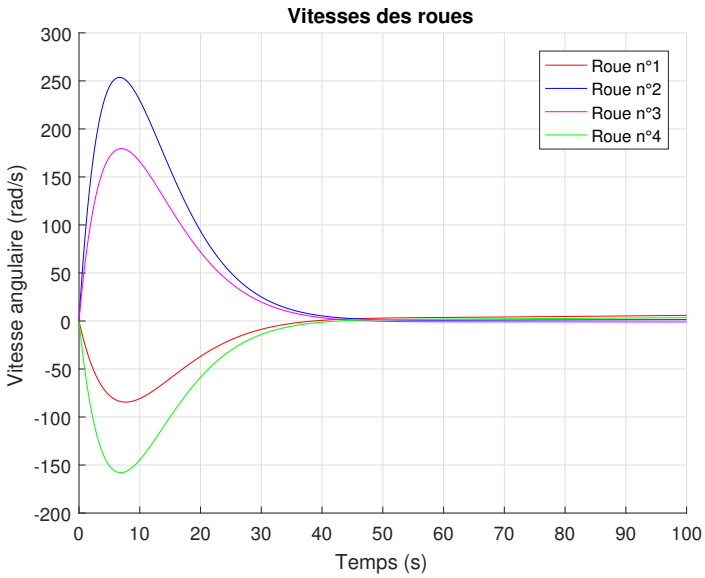

Figure 6.16 Vitesses de rotation des roues - Roue Cylindrique

Nous pouvons à nouveau comparer la consommation en puissance du système optimisé et la comparer à celle du système initial (Fig. 6.17). Dans ce cas, la puissance ne change pas par rapport au système initial. Il est cependant important de noter que l'algorithme est parvenu à conserver une consommation très proche de celle du système original tout réduisant la masse des roues. On peut en conclure que cette géométrie de roue se prête moins bien à l'optimisation sous les contraintes choisies. Cela explique aussi pourquoi la première forme est plus souvent choisie, car elle permet une plus grande inertie avec une masse moindre.

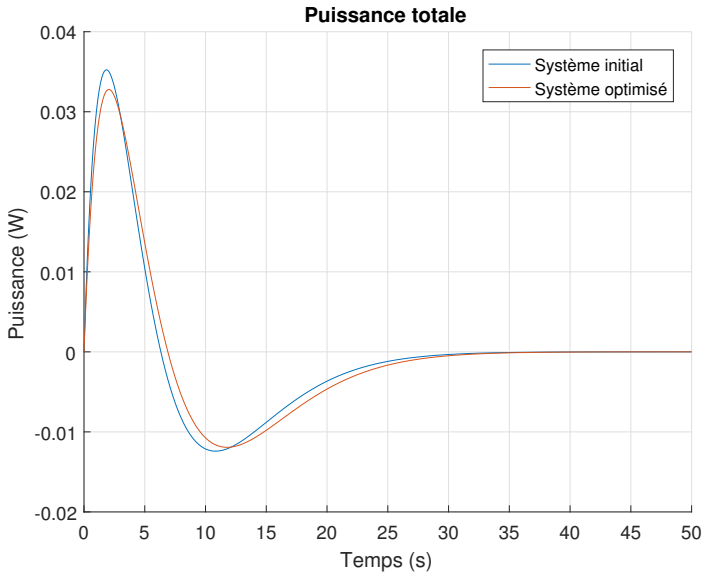

Figure 6.17 Comparaison de la consommation de puissance - Roue Cylindrique

6.1.5 Changement des contraintes

Cette partie est dédiée à représenter l'influence de la catégorisation des contraintes entre `HardGoal` et `SoftGoal`. Pour cela, la contrainte en amplitude de commande `CommandReq` est ici placée en `HardGoal`. La seconde géométrie de la roue est ici conservée et les nouveaux résultats obtenus sont détaillés ci-dessous (Tab. 6.7).

Dans ce cas, nous pouvons voir que l'algorithme n'est pas parvenu à atteindre la valeur de 25 g pour la roue, la valeur en est cependant assez proche. Le transfert de la contrainte de `SoftGoal` à `HardGoal` met sur un «pied d'égalité» les deux contraintes, et il semble que les deux ne soient pas atteignables en même temps. Ceci justifie que la masse obtenue finalement ne soit pas celle imposée.

Tableau 6.7 Dimensions géométriques de la roue n°2 obtenues par co-design

Paramètre	Valeur	Unité
r_p	0	mm
r_g	13.3	mm
h	0	mm
H	6	mm
m	$26.2.10^{-3}$	kg
J_w	3.8012×10^{-6}	kg.m^2

On obtient également des valeurs pour les angles de la structure pyramidale soutenant les roues (Tab. 6.8)

Tableau 6.8 Angles de la structure pyramidale avec roues cylindriques

Paramètre	Valeur	Unité
a	0.1	
b	0.23	
α	11.42	°
β	25.9	°

Simulations

Les gains du correcteur retournés par `systune` sont :

$$\begin{aligned} \mathbf{K}_p =& 10^{-3} \times \begin{bmatrix} 0.9852 & 0 & 0 \\ 0 & 0.9838 & 0 \\ 0 & 0 & 0.2046 \end{bmatrix} \\ \mathbf{K}_d =& \begin{bmatrix} 0.0063 & 0 & 0 \\ 0 & 0.0062 & 0 \\ 0 & 0 & 0.0012 \end{bmatrix} \end{aligned} \tag{6.13}$$

Les valeurs sont cette fois-ci bien différentes des valeurs initiales. On trace à nouveau les réponses temporelles pour la deuxième géométrie de roue pour la même commande (Figs. 6.18, 6.19, 6.20 et 6.21). Une nouvelle fois les temps de réponse sont bien respectés.

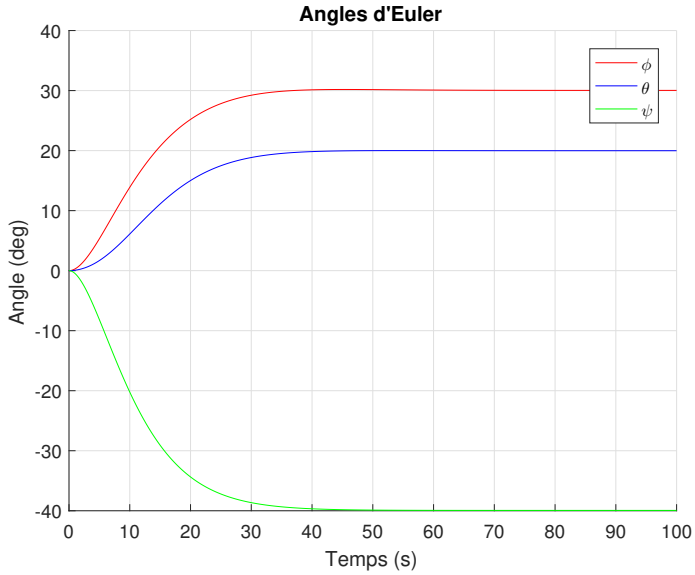

Figure 6.18 Réponses temporelles - Changement des contraintes

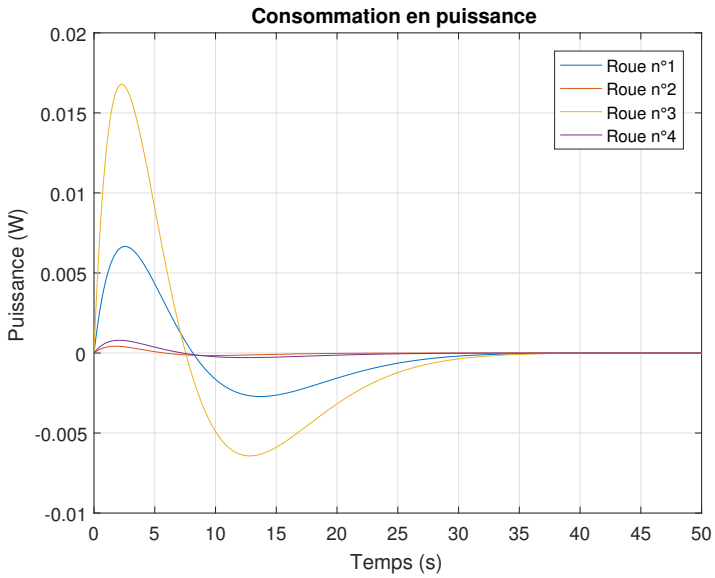

Figure 6.19 Consommation en puissance des 4 roues - Changement des contraintes

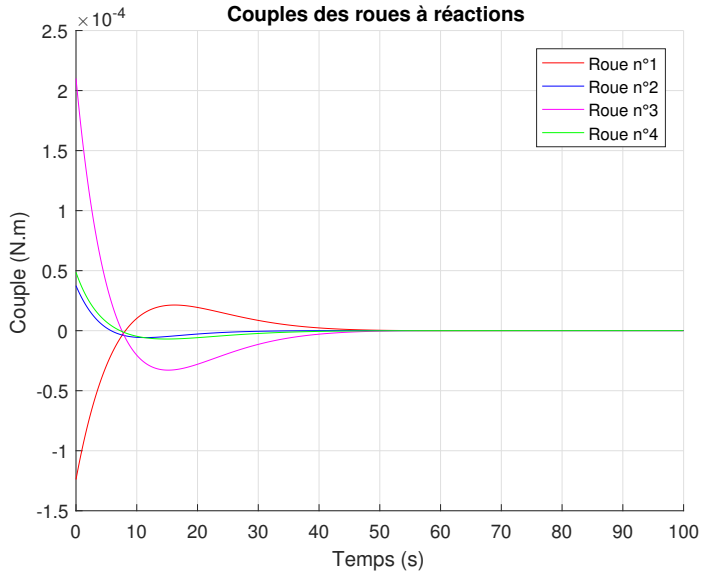

Figure 6.20 Couples des roues - Changement des contraintes

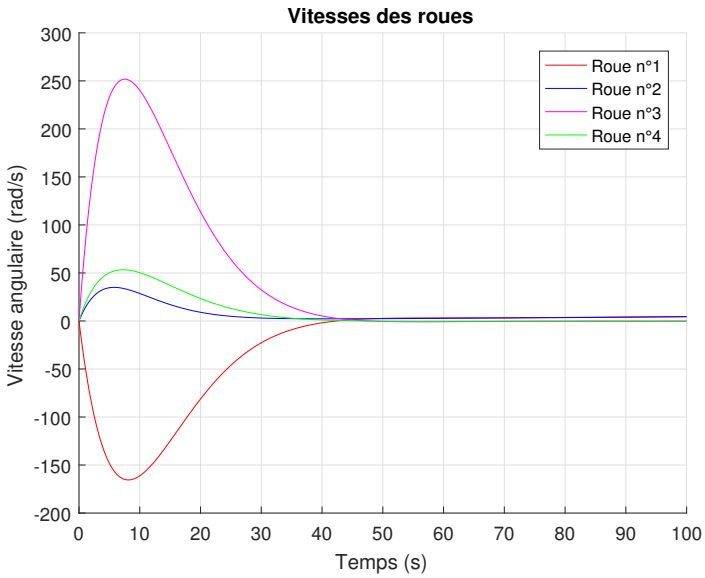

Figure 6.21 Vitesses de rotation des roues - Changement des contraintes

Afin de conclure quant aux performances de l'algorithme pour le co-design des roues de réaction, nous pouvons comparer les différentes consommations de puissance entre le système initial, et les systèmes optimisés avec la contrainte `CommandReq` placée en `SoftGoal` ou en `HardGoal`.

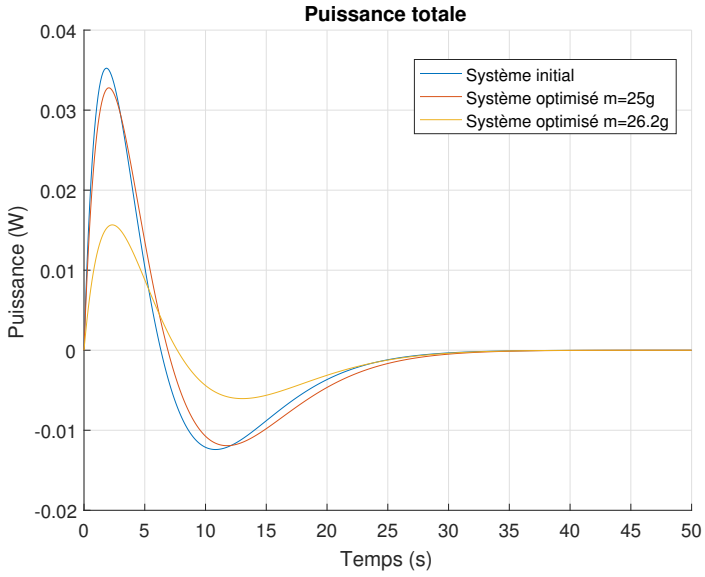

Figure 6.22 Comparaison de la consommation de puissance entre le système initial et les deux optimisations

On peut noter que l'algorithme propose deux systèmes différents selon la position des contraintes en tant que `SoftGoal` ou `HardGoal`. En effet, si des concessions sont faites sur la contrainte en masse maximale des roues au profit de l'atténuation de la consommation en puissance, la fonction `systune` permet d'obtenir un bon compromis entre les deux contraintes. La masse finale est ainsi proche de la valeur souhaitée (26.2 g contre 25 g) et la puissance est nettement moindre.

On peut comprendre ici tout l'intérêt que le co-design pourrait apporter lors des étapes de conception de systèmes de commande. En effet, le procédé standard largement répandu est composé de deux étapes : l'optimisation mécanique puis celle du correcteur à partir des contraintes du cahier des charges permettant d'obtenir par exemple le système initial défini dans ce chapitre. Toutefois, le co-design avec `systune` permet de réaliser ces deux tâches en parallèle dans le but d'obtenir des performances encore supérieures selon les contraintes imposées comme le prouve la figure 6.22.

Nous pouvons donc conclure que le co-design présente un fort potentiel lors de la conception des systèmes de commande, et ce dès les phases de conceptions préliminaires. Cela permet

en effet d'obtenir des performances supérieures en comparaison des méthodes standards couramment appliquées dans l'industrie.

6.2 Optimisation du système de désaturation

La phase de désaturation intervient lorsque les roues de réaction approchent de leur limite en vitesse de rotation (on dit qu'elles saturent). Il n'est alors plus possible de continuer à les accélérer et donc de contrôler l'attitude du satellite. Il est alors nécessaire de les ralentir afin de pouvoir continuer à assurer l'orientation du satellite. Toutefois il n'est pas possible de simplement inverser la commande aux bornes du moteur des roues, car cela modifierait l'attitude en conséquence. Afin de conserver la même orientation lors de cette phase de décélération, une solution courante est de compenser le couple induit par le freinage des roues en utilisant des magnéto-coupleurs. C'est ce même mécanisme qui est appelé désaturation des roues de réaction.

La dynamique associée aux magnéto-coupleurs a déjà été abordée dans le Chapitre 4. Cette partie-ci se concentre sur l'implantation dans MATLAB de leurs géométries et fonctionnement afin de les optimiser comme cela a été fait avec les roues de réaction. Nous utilisons dans la suite les résultats de la section précédente pour la géométrie de roues dite creuses et allons utiliser `systune` pour optimiser le mécanisme de désaturation des roues obtenues ainsi que les bobines associées.

6.2.1 Paramétrisation des bobines

Afin de pouvoir optimiser la géométrie des bobines, il faut dans un premier temps relier le moment qu'elles exercent sur le satellite à leurs dimensions spatiales. Il est ainsi possible de relier le dipôle magnétique d'une bobine à l'intensité I ou la tension V appliquée à ses bornes et ses grandeurs géométriques selon la formule :

$$m = KNIA \tag{6.14}$$

Ou en appliquant la loi d'Ohm :

$$m = \frac{KNA}{R}V = MV \tag{6.15}$$

avec N le nombre de tours du fil de cuivre constituant la bobine, A l'aire de la boucle du courant, R la résistance du fil et K un coefficient symbolisant l'effet d'un cœur magnétique à l'intérieur de la bobine.

L'expression de la résistance électrique R du fil est obtenue comme indiqué dans [17] par :

$$R = \frac{l_w.R_{Cu}}{a_w} \tag{6.16}$$

où l_w est la longueur totale du fil enroulé, $R_{Cu} = 1,55.10^{-8}\,\Omega$.m la résistivité du cuivre et $a_w = 7,97.10^{9}\,\text{mm}^2$ le calibre du fil. De plus, le coefficient K s'exprime par la formule suivante :

$$K = \begin{cases} 1 & \text{sans cœur} \\ 1 + \frac{\mu_r - 1}{1+(\mu_r-1)N_d} & \text{avec un cœur de perméabilité } \mu_r \end{cases} \tag{6.17}$$

Dans notre cas, nous considérerons un cœur de perméabilité $\mu_r = 2000$. Le facteur N_d dépend de la forme et taille du cœur magnétique ; celui-ci sera détaillé plus loin dans le chapitre.

Le satellite étudié dans ce chapitre est de type Cubesat 3U (3 Unités). Pour ce type de satellite, le système d'orientation magnétique est souvent composé de 2 bobines cylindriques avec un cœur magnétique (selon les axes $\mathbf{x}_b$ et $\mathbf{y}_b$ du satellite) et d'une bobine carrée sans cœur selon l'axe $\mathbf{z}_b$. Cette disposition permet d'avoir un système compact tout en garantissant un contrôle précis dans toutes les directions.

Les deux géométries de bobine sont ici étudiées séparément dans le but d'obtenir la paramétrisation qui permettra de réaliser l'optimisation du système dans sa totalité.

Bobine carrée

Cette bobine n'a pas de cœur magnétique, le facteur K est donc égal à 1. De plus, celle-ci est formée par un enroulement de N_{ca} spires de forme carrée de côté c. L'aire du carré décrite par les spires est donnée par :

$$A_{ca} = c^2 \tag{6.18}$$

On obtient ensuite la longueur totale du fil en fonction :

$$l_{w_{ca}} = 4cN_{ca} \tag{6.19}$$

ainsi que la masse de la bobine carrée :

$$m_{ca} = \rho_{Cu} a_w l_{w_{ca}} \tag{6.20}$$

avec $\rho_{Cu} = 8.93 \times 10^3\,\text{kg.m}^{-3}$, selon les données fournies par PolyOrbite.

Les grandeurs géométriques de la bobine sont définies dans le tableau 6.9. Les limites sont

fixées par comparaison avec les magnéto-coupleurs disponibles sur le marché des Cubesats et tient compte que la bobine ne peut pas dépasser 10 cm de longueur (dimension latérale du Cubesat).

Tableau 6.9 Paramétrisation de la bobine carrée

Paramètre	Valeur initiale	Minimum	Maximum
N_{ca}	1000	500	100000
c	0.07	0.05	0.1

Dans MATLAB , ces paramètres sont définis par les fonctions :

```
N_ca     = realp('N_ca',1000);
N_ca.min = 500;
N_ca.max = 100000;

c     = realp('c',0.07);
c.min = 0.05;
c.max = 0.1;
```

On utilise ensuite ceux-ci pour construire les grandeurs qui serviront à modéliser le fonctionnement de la bobine en tant que magnéto-coupleur ainsi que sa masse dans le but de la minimiser. Ces formules sont détaillées plus haut dans cette section.

```
R_ca = 4*c*N_ca*Rcu/aw;
M_ca = N_ca*c^2/R_ca;
m_ca = N_ca*aw*4*c*rho_cu;
```

Bobines cylindriques avec cœur magnétique

Pour un cylindre de rayon r, de longueur l et de nombre de tours de fil N_{cyl}, on peut exprimer l'aire d'une section par la formule :

$$A_{cyl} = \pi r^2 \tag{6.21}$$

ainsi que de la longueur totale du fil enroulé autour du cylindre par :

$$l_{w_{cyl}} = 2\pi r N_{cyl} \tag{6.22}$$

La masse totale de la bobine est alors la somme du cœur magnétique et du fil enroulé autour :

$$m_{cyl} = \rho_{cœur}\pi r^2 l + \rho_{Cu} a_w l_{w_{cyl}} \tag{6.23}$$

avec $\rho_{cœur} = 8.74 \times 10^3$ kg.m^{-3} pour un cœur en ferrite de fer, selon les données fournies par PolyOrbite. De plus, le coefficient N_d présent dans la formule du coefficient K se calcule par la formule [17] :

$$N_d = \frac{4[\ln\left(\frac{l}{r}\right) - 1]}{\left(\frac{l}{r}\right)^2 - 4\ln\left(\frac{l}{r}\right)} \tag{6.24}$$

Comme cela va être détaillé dans la suite, nous allons paramétrer de façon différente la longueur l et le rayon r. Les paramètres qui seront choisis comme variables seront les variations par rapport aux valeurs moyennes l_m et r_m, telles que définies ci-dessous :

$$\begin{cases} l = & l_m + \Delta l \\ r = & r_m + \Delta r \end{cases} \text{ avec } r_m = 0.005 \text{ et } l_m = 0.05 \tag{6.25}$$

Les grandeurs géométriques de la bobine sont définis dans le tableau 6.10.

Tableau 6.10 Paramétrisation de la bobine cylindrique

Paramètre	Valeur initiale	Minimum	Maximum
N_{cyl}	1000	500	100000
Δl	0.01	-0.04	0.04
Δr	0.001	-0.003	0.003

Ce qui dans MATLAB se construit de la façon suivante :

```
N_cyl=realp('N_cyl',1000);
N_cyl.min=500;
N_cyl.max=100000;

rm=0.005;
Dr=realp('Dr',0.001);
Dr.min=-0.003;
Dr.max=-0.003;

lm=0.05;
Dl=realp('Dl',0.01);
Dl.min=-0.04;
```

```
Dl.max=0.04;

%Parametres longueur
l=lm+Dl;
r=rm+Dr;
```

Paramétrisation du logarithme

Lors de la définition de N_d un problème se pose : il n'est pas possible d'utiliser la fonction ln sur un paramètre `realp`. Une piste proposée est d'utiliser une décomposition en série de Taylor du logarithme en laissant l'ordre d'approximation variable sachant que plus il est grand, plus le temps de calcul de `systune` sera long.

On rappelle ici la formule de la série de Taylor pour le logarithme népérien :

$$\ln(1+x) \approx x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots \tag{6.26}$$

Toutefois cette approximation n'est valable que pour $|x| < 1$, il nous faut réécrire la formule de N_d en modifiant l'expression du logarithme népérien :

$$\ln\left(\frac{l}{r}\right) = \ln\left(\frac{l_m + \Delta l}{r_m + \Delta r}\right) = \ln\left(\frac{l_m}{r_m} \times \frac{1 + \frac{\Delta l}{l_m}}{1 + \frac{\Delta r}{r_m}}\right) \tag{6.27}$$

On peut également écrire :

$$\ln\left(\frac{l}{r}\right) = \ln\left(\frac{l_m}{r_m}\right) + \ln\left(1 + \frac{\Delta l}{l_m}\right) - \ln\left(1 + \frac{\Delta r}{r_m}\right) \tag{6.28}$$

En utilisant l'approximation des séries de Taylor, on obtient alors :

$$\ln\left(\frac{l}{r}\right) \approx \ln\left(\frac{l_m}{r_m}\right) + \frac{\Delta l}{l_m} - \frac{\Delta r}{r_m} - \frac{\left(\frac{\Delta l}{l_m}\right)^2}{2} + \frac{\left(\frac{\Delta r}{r_m}\right)^2}{2} + \cdots \tag{6.29}$$

Après plusieurs essais, il est apparu qu'un ordre $k = 4$ était suffisant pour approcher la valeur du logarithme sans trop allonger le temps de calcul. Le code MATLAB utilisé pour calculer N_d prend donc la forme :

```
k   = 4;
lnl = 0; %% initialisation
lnr = 0;
```

```
for i=1:k
    lnl=lnl+((-1)^(i+1)*(Dl/lm)^i/i);
    lnr=lnr+((-1)^(i+1)*(Dr/rm)^i/i);
end

ln = log(lm/rm)+lnl-lnr;
Nd = 4*((ln-1)/((l/r)^2-4*ln));
```

Nous pouvons finalement paramétrer les grandeurs associées aux bobines cylindriques de la façon suivante :

```
R_cyl = 2*pi*r*N*Rcu/aw;
lwc   = 2*pi*r*N;
M_cyl = pi*r^2*N*(1+((mur-1)/(1+(mur-1)*Nd)))/R;
m_cyl = rhocore*pi*r^2*l+aw*lwc*rhocu;
```

Le Cubesat comporte 2 bobines cylindriques selon les axes $\mathbf{x}_b$ et $\mathbf{y}_b$ et une bobine carrée selon l'axe $\mathbf{z}_b$. Il est donc possible de relier le moment magnétique des bobines $\mathbf{m}$ en fonction de la tension $\mathbf{V}$ appliquée aux bornes de celles-ci par la relation :

$$\mathbf{m} = \mathbf{MV} = \begin{bmatrix} M_{cyl} & 0 & 0 \\ 0 & M_{cyl} & 0 \\ 0 & 0 & M_{ca} \end{bmatrix} \mathbf{V} \tag{6.30}$$

6.2.2 Architecture du correcteur

À partir du système défini dans le chapitre précédent, le mécanisme de désaturation des roues par les magnéto-coupleurs est ajouté dans le code MATLAB . Les équations permettant de modéliser ce mode de fonctionnement ainsi que le correcteur associé sont données dans cette section et dans le Chapitre 4. Le but est ici de réduire les moments angulaires $\mathbf{h}$ des roues jusqu'aux valeurs souhaitées $\mathbf{h_r}$ sans impacter l'attitude du satellite. L'architecture du correcteur est donnée à la figure 6.23 ; les blocs en rouge sont ceux qui sont optimisés par `systune` .

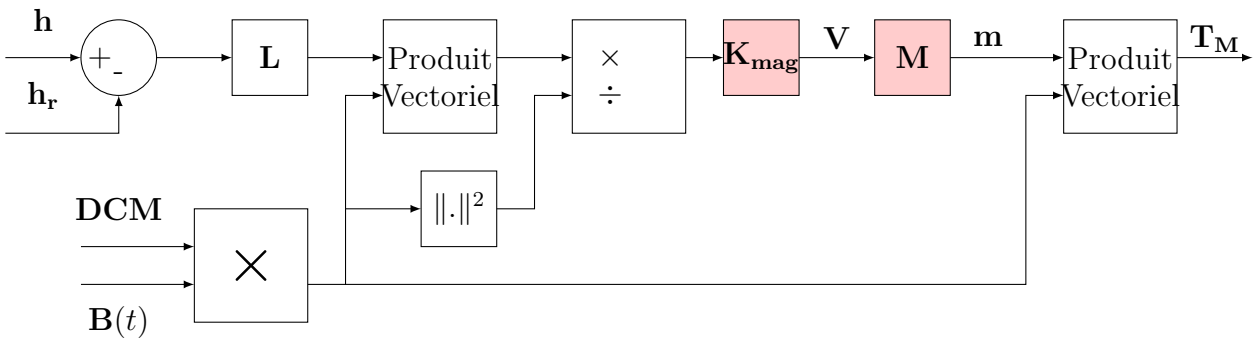

Figure 6.23 Architecture du correcteur des magnéto-coupleurs

Le bloc **M** a déjà été décrit plus tôt dans la section, et on définit le gain $\mathbf{K_{mag}}$ de la façon suivante :

```
Kmag              = realp('Kmag',0.1*eye(3));
Kmag.Minimum      = 0.00001*eye(3);
Kmag.free(1,2) = false;
Kmag.free(1,3) = false;
Kmag.free(2,1) = false;
Kmag.free(2,3) = false;
Kmag.free(3,1) = false;
Kmag.free(3,2) = false;
```

Le gain est ici minoré, car celui-ci doit être positif par définition de la loi de commande. Cela nous pousse donc à imposer cette limite afin de faciliter la convergence de `systune` .

6.2.3 Contraintes

Réponse temporelle

Le but est ici de parvenir à réduire la vitesse de rotation des roues. Dans la plupart des cas, cela est réalisé sur une durée s'étalant sur plusieurs orbites afin de respecter les limites des magnéto-coupleurs. Nous appliquons une nouvelle fois une contrainte en `StepTracking` entre le moment angulaire des roues commandé $\mathbf{h_r}$ et leur moment angulaire effectif **h**. Cependant, cette fois-ci la réponse imposée est celle d'un ordre 1.

```
Tr = 10000;
TrackReq = TuningGoal.StepTracking({'h1r','h2r','h3r','h4r'},{'h1'
    ,'h2','h3','h4'},Tr);
```

```
TrackReq.RelGap = 0.05;
```

Il est nécessaire que cette contrainte soit définie en tant que `HardGoal` afin que `systune` optimise $\mathbf{K_{mag}}$ en tenant compte du temps désiré.

Puissance consommée

Afin de limiter la puissance requise par les bobines, nous pouvons imposer deux contraintes réduisant les résistances des deux types de magnéto-coupleurs.

```
R_Req1 = TuningGoal.Gain({'V1'},{'P1'},100);
R_Req2 = TuningGoal.Gain({'V2'},{'P2'},100);
```

Ces deux contraintes de conception ont été placées en `HardGoal` dans la suite des travaux afin d'assurer la minimisation de la consommation de puissance.

Masses des bobines

En plus de réduire la résistance des bobines, le cahier des charges requiert également de minimiser la masse de celles-ci. Comme cela a été fait lors de l'optimisation des roues, cela requiert des contraintes en gains telles que celles-ci :

```
m_Req1 = TuningGoal.Gain({'m1in'},{'m1out'},0.03);
m_Req2 = TuningGoal.Gain({'m2in'},{'m2out'},0.05);
```

Dans la suite des travaux, ces contraintes sont considérées comme moins importantes

Amplitude de la commande en tension

Les magnéto-coupleurs ne peuvent supporter de trop grandes tensions. Dans le cas d'un Cubesat, la limite est usuellement de 5V. Afin de tenir compte d'une telle limitation, et de limiter l'amplitude de la commande des bobines lors de la désaturation des roues, une contrainte est également définie entre la commande $\mathbf{h_r}$ et la tension appliquée aux bornes des bobines $\mathbf{V}$.

```
V_Req = TuningGoal.Gain({'h1r','h2r','h3r','h4r'},{'Vx','Vy','Vz'
    },0.6);
```

Par expérience, il apparaît que cette contrainte est plus efficace si elle est définie en tant que `HardGoal`, permettant de réduire davantage l'amplitude de $\mathbf{V}$.

Conservation de l'attitude

Enfin, dans le but de garantir que la décélération des roues n'affectera pas le maintien en attitude du satellite, nous introduisons une autre contrainte en `StepRejection`.

```
WT = TuningGoal.StepRejection({'h1r','h2r','h3r','h4r'},{'phi','
    theta','psi'},0.001,1000);
```

Celle-ci est définie en `SoftGoal` afin de ne pas altérer le respect des autres contraintes.

6.2.4 Résultats

Il est important de noter ici que même après avoir défini ainsi le système variable et les contraintes, `systune` ne parvient pas toujours à stabiliser le système au cours des itérations. Il est donc nécessaire d'exécuter l'algorithme à partir de différentes valeurs aléatoires (dont certaines ne convergeront pas) afin d'obtenir de meilleurs résultats.

Système optimisé

Cette partie rassemble les valeurs retournées par `systune` .

La valeur optimisée du gain $\mathbf{K_{mag}}$ de désaturation est :

$$\mathbf{K_{mag}} = \begin{bmatrix} 0.0219 & 0 & 0 \\ 0 & 0.0222 & 0 \\ 0 & 0 & 0.0159 \end{bmatrix} \tag{6.31}$$

De même, les gains du correcteur PD des roues de réaction pour ce mode de fonctionnement sont :

$$\begin{aligned} \mathbf{K}_p &= \begin{bmatrix} 5.3829 & 0 & 0 \\ 0 & 2.0738 & 0 \\ 0 & 0 & 3.4620 \end{bmatrix} \\ \mathbf{K}_d &= \begin{bmatrix} 1.8097 & 0 & 0 \\ 0 & 1.4925 & 0 \\ 0 & 0 & 1.8719 \end{bmatrix} \end{aligned} \tag{6.32}$$

Pour leur part, les grandeurs géométriques des bobines sont données dans les tableaux 6.11 (bobines cylindriques) et 6.12 (bobine carrée). La masse totale du magnéto-coupleur cylindrique obtenu, constitué du cœur ferromagnétique et de la bobine, est de 31g ce qui est peu et tout à fait comparable aux magnéto-coupleurs vendus pour les CubeSats.

Tableau 6.11 Optimisation de la bobine cylindrique

Paramètre	Valeur optimisée	Unité
N_{cyl}	7250	tours
Δl	0.0262	m
Δr	-0.00187	m
l	0.0762	m
r	0.0031	m
m_{cyl}	31	g

Tableau 6.12 Optimisation de la bobine carrée

Paramètre	Valeur optimisée	Unité
N_{ca}	3260	tours
c	0.0682	m
m_{ca}	63.3	g

Le magnéto-coupleur carré obtenu par `systune` est de masse 63.3 g ce qui est une masse embarquée raisonnable pour une bobine. Cependant il est possible de trouver ce type de magnéto-coupleurs sur le marché avec une masse de 50 g, ce qui prouve que l'on pourrait chercher à optimiser davantage la géométrie de cette bobine.

Simulation de la désaturation

Nous allons simuler ici la désaturation de roues à 6000 tr/min afin de confirmer que les valeurs obtenues ci-dessus respectent les contraintes imposées. En premier lieu, on étudie l'évolution de la vitesse des roues de réactions en figure 6.24. Le but est de les ralentir aux alentours de 0 afin de revenir au mode de pointage précis du Cubesat. On peut voir ici que les roues atteignent bien 0 tr/min entre 5000 et 6000 s. Cette durée correspond environ à une période orbitale 5577.6 s.

Nous pouvons ensuite vérifier que l'attitude du satellite demeure inchangée lors de cette opération sur la figure 6.25. On constate bel et bien que les angles ϕ, θ et ψ décrivant l'attitude du Cubesat ne changent que très peu (de l'ordre de 10^{-5} °).

De même la tension de commande des magnéto-coupleurs ne dépasse pas 5 V, qui est généralement la limite de ce genre de système embarqué. On peut voir les commandes séparées des 3 actionneurs magnétiques sur la figure 6.26.

Enfin, la puissance consommée pour maintenir l'orientation du satellite tandis que les roues ralentissent demeure ici inférieure à 0.02 W. De façon générale, la puissance maximale associée

aux magnéto-coupleurs pour un Cubesat est d'environ 200 mW donc 10 fois supérieure à la puissance nécessaire ici.

En conclusion, nous sommes une nouvelle fois parvenus à optimiser le correcteur et les actionneurs pour un mode de fonctionnement différent. Cela met bien en avant les possibilités offertes par `systune` en matière de co-design. L'algorithme s'avère être polyvalent et adaptable à de nombreux systèmes, tant que ceux-ci sont correctement définis dans MATLAB.

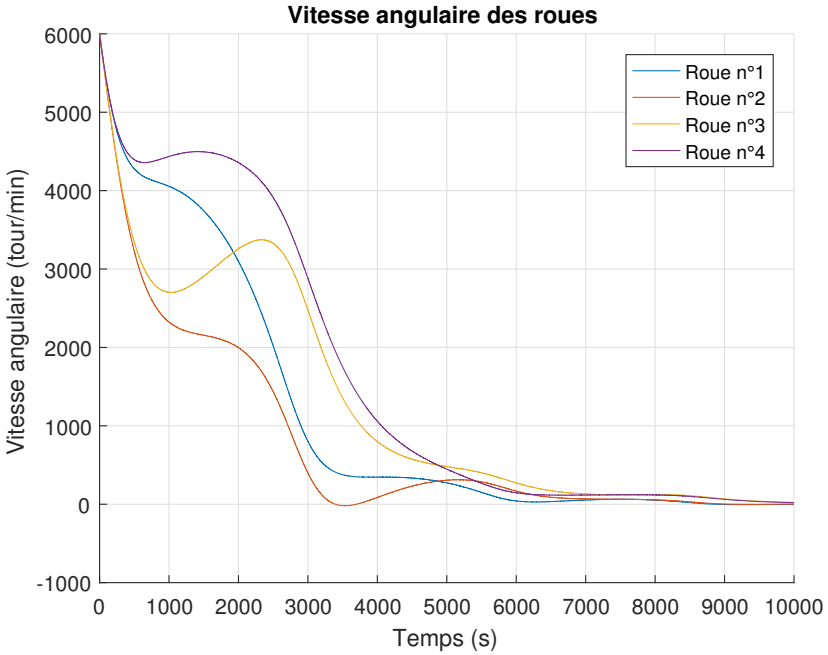

Figure 6.24 Désaturation des roues

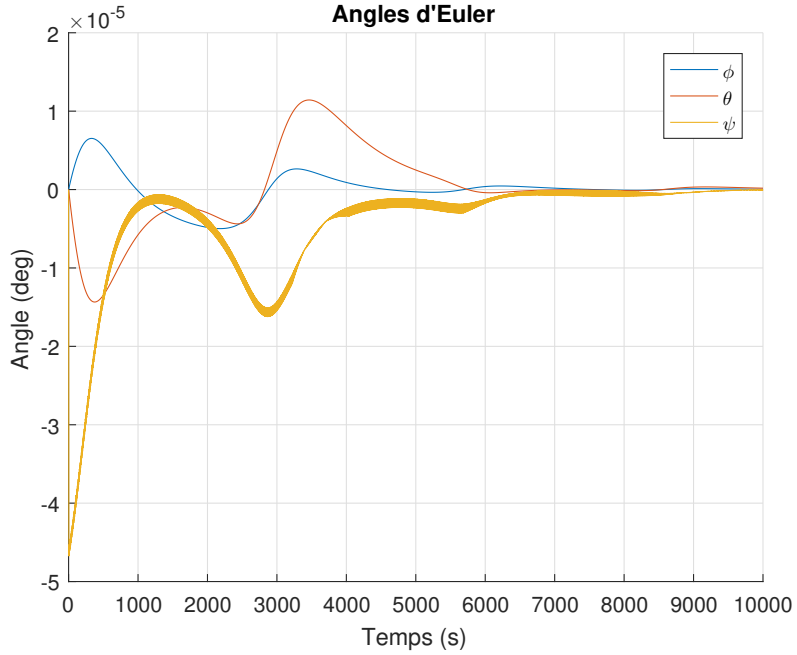

Figure 6.25 Variation de l'attitude au cours de la désaturation des roues

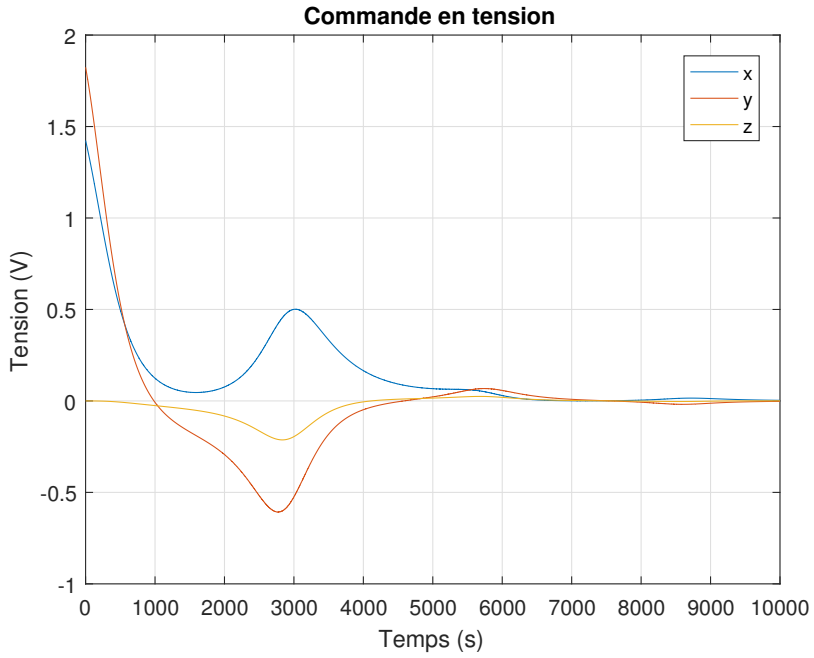

Figure 6.26 Commandes des magnéto-coupleurs

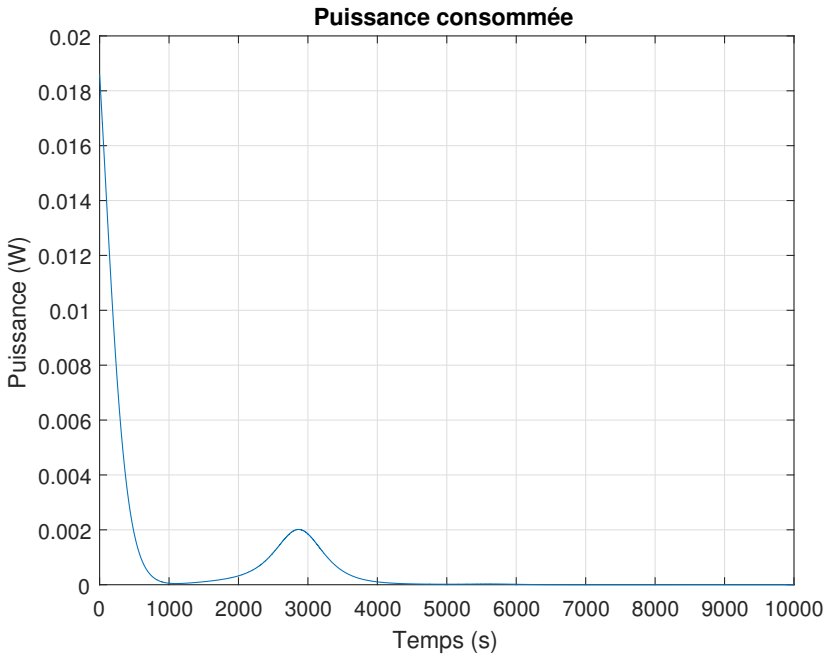

Figure 6.27 Puissance consommée par la désaturation

CHAPITRE 7 CONCLUSION

7.1 Résultats et discussions

L'objectif principal de ces travaux de recherche était de développer un procédé permettant d'optimiser les actionneurs d'un nanosatellite en parallèle de sa loi de commande grâce à la fonction `systune` de MATLAB .

Le premier objectif de ces travaux était de parvenir à utiliser la synthèse $\mathcal{H}_\infty$ structurée pour optimiser le système de pointage d'un satellite constitué de roues de réaction. Cette partie a donc permis de développer une méthode simple présentant les étapes nécessaires à la réalisation du co-design d'un système. Dans un premier temps, nous avons paramétrés les actionneurs ainsi que les gains du correcteur sous la forme de coefficients variables `realp`. Ceux-ci ont ensuite été incorporés dans la modélisation du satellite. Il a enfin été nécessaire de définir des contraintes correspondant au cahier des charges des missions du satellite, tel que le temps de réponse ou la masse des roues de réactions, qui ont été implémentées dans `systune` . Afin de confirmer que l'algorithme permettait bel et bien de réaliser le co-design d'un système avec différents paramètres variables, plusieurs exécutions de celui-ci ont été réalisées. Cela a permis de mieux rendre compte du potentiel du co-design dans le domaine de l'optimisation des performances. En effet, en augmentant le nombre de paramètres variables nous sommes parvenus à améliorer les performances du satellite (dont la puissance consommée) alors que le système initial était pourtant considéré comme optimisé par une étude antérieure. Nous avons donc prouvé que le co-design permettait d'atteindre des performances supérieures par rapport aux méthodes dites standard.

Le second objectif de ce mémoire a ensuite été d'appliquer le procédé précédemment défini au cas d'un nanosatellite de format Cubesat en cours de fabrication par la société technologique PolyOrbite.Nous avons ensuite effectué le co-design pour deux phases de fonctionnement distinctes : le pointage précis selon une direction avec 4 roues de réactions et la désaturation de ces roues à l'aide de 3 magnétocoupleurs. Afin d'aider à la conception de ces actionneurs et de l'algorithme de commande du satellite, les dimensions respectives de ceux-ci ont été paramétrées. Cela a permis d'obtenir, grâce à `systune` , un système plus performant répondant au cahier des charges défini par PolyOrbite. Deux types de roues ont été optimisées afin de comparer leurs performances. Dans chacun des cas, une roue plus légère ainsi qu'un système moins énergivore respectant toutes les contraintes imposées a été obtenu. De plus, nous avons testé différentes hiérarchies de contraintes pour rendre compte de la flexibilité et des possibilités offertes par `systune` dans l'optimisation des systèmes. Nous avons ainsi

pu voir que selon la traduction du cahier des charges dans l'interface MATLAB les performances du correcteur en attitude pouvaient être améliorées de différentes manières. Enfin, l'algorithme a été appliqué au mode de désaturation des roues précédemment optimisées. Les magnéto-coupleurs, constitués de bobines, ont été paramétrés de telles sortes que leurs moments magnétiques, masses et résistances soient exprimés sous la forme d'opérations simples de paramètres `realp`. Les contraintes de conception ont ensuite été implantées dans le but de tester `systune` pour un mécanisme différent du satellite. Cela a permis de confirmer que l'algorithme fonctionnait également pour co-dimensionner les bobines du Cubesat et par extension que la méthode développée ici pourrait être appliquée à d'autres types de systèmes. Nous avons également pu conclure que le co-design permettait d'obtenir des performances encore supérieures à celles qui étaient obtenues par les méthodes de conception standard.

7.2 Limitations et améliorations futures

Les travaux développés dans ce mémoire proposent une démarche pour réaliser une tâche longtemps considérée comme complexe et laborieuse. En effet, si l'optimisation de plusieurs paramètres en simultané a toujours été un défi, la fonction `systune` promet à présent de simplifier les étapes du co-design et avant tout de pouvoir améliorer significativement les performances des systèmes. En permettant d'imposer des contraintes mécaniques (masse, volume ...) et en réponse à la commande (temporelle, atténuation de vibration...), la synthèse $\mathcal{H}_\infty$ structurée pourrait donc devenir une étape incontournable lors de la conception préliminaire des systèmes. Celle-ci permettrait en effet de nombreuses révisions entre les équipes de conception mécanique et les ingénieurs en commande des systèmes en effectuant les deux tâches à la fois.

Néanmoins, la méthode définie dans ce mémoire reste à perfectionner. En effet, une attention particulière doit être portée lors de la paramétrisation des actionneurs et des gains ainsi que lors de la définition des contraintes dans MATLAB sous peine de ne jamais voir `systune` converger vers une solution stable au cours de ses itérations. De futurs travaux pourraient se concentrer sur l'étude des contraintes qu'il est possible d'imposer grâce à `systune` , et surtout déterminer comment choisir lesquelles sont les plus adaptées pour transcrire le cahier des charges ainsi que comment les définir en conséquence. La diversité de choix possibles a été un grand obstacle lors de la définition des systèmes lors ces travaux, et il a parfois été long de trouver les valeurs pour lesquelles l'algorithme va tendre vers les résultats attendus. De même, l'utilisation de paramètres variables `realp` contraint à ne pouvoir utiliser que des opérateurs simples tels que la somme ou la multiplication lors de la définition des grandeurs du système. Des solutions ont été apportées dans ce mémoire, par exemple l'utilisation des

formules d'angles moitié ou des séries de Taylor. Cependant, il pourrait être intéressant de développer de nouvelles méthodes pour contourner cette limitation afin de modéliser plus fidèlement la mécanique des actionneurs.

Finalement, bien que les études réalisées ici soient consacrées au cas de satellites la démarche qui a été utilisée peut être appliquée à d'autres systèmes aérospatiaux (avions, drones...) ou en général tout système pour lequel on cherche à développer un correcteur. Le co-design apparaît donc comme une méthode de conception prometteuse. La perspective d'obtenir de meilleures performances devrait permettre l'essor de nombreux travaux dans les années à venir, et donc de voir se développer cette méthode pour pouvoir l'appliquer à la conception de tout système de commande.

RÉFÉRENCES

[1] S. R. Starin et J. Eterno, "Attitude determination and control systems," 2011. [En ligne]. Disponible : https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110007876.pdf

[2] J. Schoonwinkel, "Attitude determination and control system of a nanosatellite," Mémoire de maîtrise, Stellenbosch : University of Stellenbosch, 2007.

[3] R. Votel et D. Sinclair, "Comparison of control moment gyros and reaction wheels for small earth-observing satellites," dans *Proceedings of the AIAA/USU Conference on Small Satellites*, 2012. [En ligne]. Disponible : https://digitalcommons.usu.edu/smallsat/2012/all2012/74/

[4] A. Bellar, M. A. S. Mohammed et A. Adnane, "Minimum power consumption of the microsatellite attitude control using pyramidal reaction wheel configuration," dans *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*. Algiers, Algeria : IEEE, 2016, p. 253–257.

[5] C.-H. Won, "Comparative study of various control methods for attitude control of a leo satellite," *Aerospace science and technology*, vol. 3, n°. 5, p. 323–333, 1999.

[6] G. Zames, "Feedback and optimal sensitivity : Model reference transformations, multiplicative seminorms, and approximate inverses," *IEEE Transactions on automatic control*, vol. 26, n°. 2, p. 301–320, 1981.

[7] J. C. Doyle, K. Glover, P. P. Khargonekar et B. A. Francis, "State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems," *IEEE Transactions on Automatic control*, vol. 34, n°. 8, p. 831–847, 1989.

[8] J.-F. Magni, "An LFT approach to robust gain scheduling," dans *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, p. 7971–7976.

[9] D. Alazard, T. Loquen, H. De Plinval et C. Cumer, "Avionics/control co-design for large flexible space structures," dans *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 4638.

[10] P. Gahinet et P. Apkarian, "Structured $\mathcal{H}_\infty$ synthesis in matlab," *IFAC Proceedings Volumes*, vol. 44, n°. 1, p. 1435–1440, 2011.

[11] M. H. Kaplan, "Modern spacecraft dynamics and control," *New York, John Wiley and Sons, Inc., 1976. 427 p.*, 1976.

[12] B. N. Agrawal, *Design of geosynchronous spacecraft*. Prentice-Hall, Inc., 1986.

[13] J. R. Wertz, *Spacecraft attitude determination and control*. Springer Science & Business Media, 2012, vol. 73.

[14] C. Toglia, P. Pavia, G. Campolo, D. Alazard, T. Loquen, H. de Plinval, C. Cumer, M. Casasco et L. Massotti, "Optimal co-design for earth observation satellites with flexible appendages," dans *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 4640.

[15] B. Wie, "Dynamic modeling and attitude control of solar sail spacecraft : part i," dans *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4572.

[16] E. Bender, "An analysis of stabilizing 3U cubesats using gravity gradient techniques and a low power reaction wheel," 2011. [En ligne]. Disponible : https://digitalcommons.calpoly.edu/aerosp/41

[17] J. Li, M. Post, T. Wright et R. Lee, "Design of attitude control systems for cubesat-class nanosatellite," *Journal of Control Science and Engineering*, vol. 2013, p. 4, 2013.

[18] S. Ni et C. Zhang, "Attitude determination of nano satellite based on gyroscope, sun sensor and magnetometer," *Procedia Engineering*, vol. 15, p. 959–963, 2011.

[19] R. Wiśniewski et P. Kulczycki, "Slew maneuver control for spacecraft equipped with star camera and reaction wheels," *Control engineering practice*, vol. 13, n°. 3, p. 349–356, 2005.

[20] T. Michaelis, "Small satellite reaction wheel optimization," dans *Proceedings of the AIAA/USU Conference on Small Satellites*, 1990. [En ligne]. Disponible : https://digitalcommons.usu.edu/smallsat/1990/all1990/39/

[21] D. Sinclair, C. C. Grant et R. Zee, "Enabling reaction wheel technology for high performance nanosatellite attitude control," dans *Proceedings of the AIAA/USU Conference on Small Satellites*, 2007. [En ligne]. Disponible : https://digitalcommons.usu.edu/smallsat/2007/all2007/63/

[22] V. Carrara, R. H. Siqueira et D. Oliveira, "Speed and current control mode strategy comparison in satellite attitude control with reaction wheels," dans *Proceedings of the 21st Brazilian Congress of Mechanical Engineering (COBEM'11)*, 2011.

[23] V. Carrara et H. K. Kuga, "Current and speed control operating modes of a reaction wheel," dans *Applied Mechanics and Materials*, vol. 706. Trans Tech Publ, 2015, p. 170–180.

[24] V. Carrara, A. G. Da Silva et H. K. Kuga, "A dynamic friction model for reaction wheels," *Advances in the Astronautical Sciences*, vol. 145, p. 343–352, 2012.

[25] R. Wiśniewski, "Three-axis satellite attitude control based on magnetic torquing - linear optimal approach*," *IFAC Proceedings Volumes*, vol. 29, n°. 8, p. 91 – 96, 1996.

[26] Z. Tudor, "Design and implementation of attitude control for 3-axes magnetic coil stabilization of a spacecraft," Mémoire de maîtrise, Institutt for teknisk kybernetikk, 2011.

[27] J. Jin, S. Ko et C.-K. Ryoo, "Fault tolerant control for satellites with four reaction wheels," *Control Engineering Practice*, vol. 16, n°. 10, p. 1250–1258, 2008.

[28] Z. Ismail et R. Varatharajoo, "A study of reaction wheel configurations for a 3-axis satellite attitude control," *Advances in Space Research*, vol. 45, n°. 6, p. 750–759, 2010.

[29] E. Oland et R. Schlanbusch, "Reaction wheel design for cubesats," dans *2009 4th International Conference on Recent Advances in Space Technologies*. IEEE, 2009, p. 778–783.

[30] D. Miller, "Design optimization of the cadre magnetorquers," *Aerospades, University of Michigan, Ann Arbor, Michigan*, 2013.

[31] H. Bang, M.-J. Tahk et H.-D. Choi, "Large angle attitude control of spacecraft with actuator saturation," *Control engineering practice*, vol. 11, n°. 9, p. 989–997, 2003.

[32] L. O. Inumoh, N. M. Horri, J. L. Forshaw et A. Pechev, "Bounded gain-scheduled lqr satellite control using a tilted wheel," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, n°. 3, p. 1726–1738, 2014.

[33] J. Doyle, "Analysis of feedback systems with structured uncertainties," dans *IEEE Proceedings D-Control Theory and Applications*, vol. 129, n°. 6. IET, 1982, p. 242–250.

[34] P. Gahinet et P. Apkarian, "A linear matrix inequality approach to $\mathcal{H}_\infty$ control," *International journal of robust and nonlinear control*, vol. 4, n°. 4, p. 421–448, 1994.

[35] V. Preda, J. Cieslak, D. Henry, S. Bennani et A. Falcoz, "A $\mathcal{H}_\infty/\mu$ solution for microvibration mitigation in satellites : A case study," *Journal of Sound and Vibration*, vol. 399, p. 21–44, 2017.

[36] H. Lhachemi, D. Saussié et G. Zhu, "Performance enhancement of a self-scheduled longitudinal flight control system via multi-objective optimization," dans *2014 American Control Conference*. IEEE, 2014, p. 1377–1383.

[37] H. Kouhi, M. Kabganian, F. F. Saberi et M. Shahravi, "Robust control of a spin-stabilized spacecraft via a 1dof gimbaled-thruster and two reaction wheels," *ISA transactions*, vol. 66, p. 310–324, 2017.

[38] J. V. Burke, D. Henrion, A. S. Lewis et M. L. Overton, "HIFOO - a MATLAB package for fixed-order controller design and $\mathcal{H}_\infty$ optimization," *IFAC Proceedings Volumes*, vol. 39, n°. 9, 2006.

[39] P. Gahinet et P. Apkarian, "Decentralized and fixed-structure $\mathcal{H}_\infty$ control in matlab," dans *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, p. 8205–8210.

[40] T. Loquen, H. de Plinval, C. Cumer et D. Alazard, "Attitude control of satellites with flexible appendages : a structured $\mathcal{H}_\infty$ control design," dans *AIAA Guidance, Navigation, and Control Conference*, 2012, p. 4845.

[41] A. Falcoz, C. Pittet, S. Bennani, A. Guignard, C. Bayart et B. Frapard, "Systematic design methods of robust and structured controllers for satellites," *CEAS Space Journal*, vol. 7, n°. 3, p. 319–334, 2015.

[42] A. Shirazi et M. Mirshams, "Pyramidal reaction wheel arrangement optimization of satellite attitude control subsystem for minimizing power consumption," *International Journal of Aeronautical and Space Sciences*, vol. 15, n°. 2, p. 190–198, 2014.

[43] D. S. Bayard, "An optimization result with application to optimal spacecraft reaction wheel orientation design," dans *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, vol. 2. IEEE, 2001, p. 1473–1478.

[44] A. L. Hale, W. Dahl et J. Lisowski, "Optimal simultaneous structural and control design of maneuvering flexible spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 8, n°. 1, p. 86–93, 1985.

[45] N. Khot, "Structure/control optimization to improve the dynamic response of space structures," *Computational Mechanics*, vol. 3, n°. 3, p. 179–186, 1988.

[46] Y. C. Paw et G. J. Balas, "Parametric uncertainty modeling for LFT model realization," dans *2008 IEEE International Conference on Computer-Aided Control Systems*. IEEE, 2008, p. 834–839.

[47] S. Hecker, A. Varga et J.-F. Magni, "Enhanced LFR-toolbox for matlab," *Aerospace Science and Technology*, vol. 9, n°. 2, p. 173–180, 2005.

[48] H. Lhachemi, D. Saussié et G. Zhu, "A structured $\mathcal{H}_\infty$-based optimization approach for integrated plant and self-scheduled flight control system design," *Aerospace Science and Technology*, vol. 45, p. 30–38, 2015.

[49] J. A. Perez, C. Pittet, D. Alazard et T. Loquen, "Integrated control/structure design of a large space structure using structured $\mathcal{H}_\infty$ control," *IFAC-PapersOnLine*, vol. 49, n°. 17, p. 302–307, 2016.

[50] P. Apkarian et D. Noll, "Nonsmooth $\mathcal{H}_\infty$ synthesis," *IEEE Transactions on Automatic Control*, vol. 51, n°. 1, p. 71–86, 2006.

[51] F. Laliberté, "Modélisation et commande d'un drone hélicoptère tandem," Mémoire de maîtrise, École Polytechnique de Montréal, 2017.

[52] D. C. Guler et C. Hajiyev, "Singular value decomposition based satellite attitude determination using different sensor configurations," *International Journal of Metrology and Quality Engineering*, vol. 8, p. 15, 2017.

[53] C. C. Finlay, S. Maus, C. Beggan, T. Bondar, A. Chambodut, T. Chernova, A. Chulliat, V. Golovkov, B. Hamilton, M. Hamoudi *et al.*, "International geomagnetic reference field : the eleventh generation," *Geophysical Journal International*, vol. 183, n°. 3, p. 1216–1230, 2010.

[54] M. L. Psiaki, "Magnetic torquer attitude control via asymptotic periodic linear quadratic regulation," *Journal of Guidance, Control, and Dynamics*, vol. 24, n°. 2, p. 386–394, 2001.

[55] T. Inamori, N. Sako et S. Nakasuka, "Magnetic dipole moment estimation and compensation for an accurate attitude control in nano-satellite missions," *Acta Astronautica*, vol. 68, n°. 11-12, p. 2038–2046, 2011.

[56] J. Jin, S. Ko et C.-K. Ryoo, "Fault tolerant control for satellites with four reaction wheels," *Control Engineering Practice*, vol. 16, n°. 10, p. 1250–1258, 2008.

[57] J.-F. Trégouët, D. Arzelier, D. Peaucelle et L. Zaccarian, "Static input allocation for reaction wheels desaturation using magnetorquers," *IFAC Proceedings Volumes*, vol. 46, n°. 19, p. 559–564, 2013.

[58] F. Giulietti, A. A. Quarta et P. Tortora, "Optimal control laws for momentum-wheel desaturation using magnetorquers," *Journal of guidance, control, and dynamics*, vol. 29, n°. 6, p. 1464–1468, 2006.

[59] P. Daligault, P. Lallet et D. Saussié, "Integrated design of inertia wheel configuration and attitude control laws for a satellite," dans *2018 AIAA SPACE and Astronautics Forum and Exposition*, 2018, p. 5244.

ANNEXE A UTILISATION DE SYSTUNE EN CO-DESIGN

Cette Annexe est un mode d'emploi complet réalisé dans le cadre de cette maîtrise afin de guider quiconque souhaiterait apprendre à appliquer la synthèse $\mathcal{H}_\infty$ structurée au co-design d'un système. Il existe en effet très peu de travaux traitant de ce sujet et encore moins présentant la démarche à suivre ce qui a justifié la rédaction de ce document qui traite d'une application au cas simple d'un moteur.

Le but de cet exemple est d'apprendre à utiliser la fonction `systune` dans le but de synthétiser un contrôleur tout en optimisant certaines grandeurs du système mécanique.

Dans notre cas, le système considéré est un moteur que l'on désire contrôler en vitesse. On souhaite que son temps de réponse à 5% soit de 2 secondes avec un dépassement inférieur à 10%. Le but final est de trouver le moteur avec la plus grande constante de temps possible respectant le critère ci-dessus sans que la commande en entrée du moteur ne soit trop forte.

A.1 Définition du système

On commence par modéliser le moteur par une fonction de transfert d'ordre 1 :

$$H_{mot} = \frac{K}{\tau.s + 1}$$

avec $K = 1$ et $\tau = 0.5\,\text{s}$.

A.1.1 Architecture du correcteur

La structure de contrôle choisie est un contrôleur PI dont l'architecture est représentée à la figure A.1. Dans un premier temps, déterminons des gains de contrôleurs K_i et K_p sans changer la constante de temps τ de façon analytique pour les comparer aux futures simulations.

A.1.2 Calcul analytique des gains

On a :

$$\Omega = \frac{K}{1 + \tau s} U$$

$$U = \frac{K_i}{s}(\Omega_{ref} - \Omega) - K_p \Omega$$

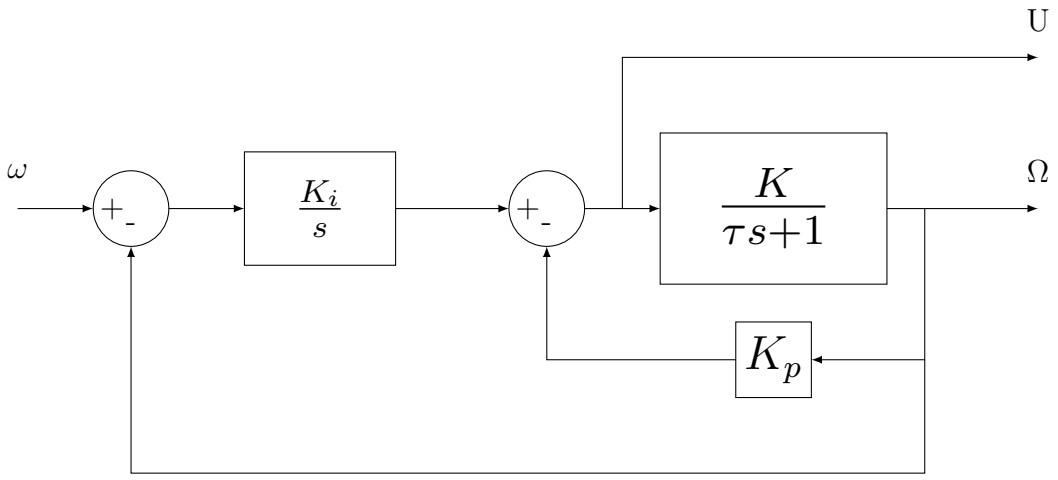

Figure A.1 Schéma de la structure de contrôle

En combinant ces deux équations, on obtient la fonction de transfert du système :

$$\frac{\Omega}{\Omega_{ref}} = \frac{KK_i/\tau}{s^2 + \left(\frac{1+KK_p}{\tau}\right)s + \frac{KK_i}{\tau}}$$

On obtient alors la pulsation naturelle ainsi que l'amortissement de ce système :

$$\omega_n = \sqrt{KK_i/\tau}$$

$$\xi = \frac{1 + KK_p}{2\sqrt{\tau KK_i}}$$

Le temps de réponse à 2% s'exprime alors comme :

$$T_r = \frac{4}{\xi\omega_n} = \frac{8\tau}{1 + KK_p}$$

Et le dépassement relatif est :

$$d = \exp\left(-\pi\frac{\xi}{\sqrt{1-\xi^2}}\right) = \exp\left(-\pi\frac{1 + KK_p}{\sqrt{4\tau KK_i - 1 - 2KK_p - K^2K_p^2}}\right)$$

Pour définir les gains du correcteur, nous nous sommes fixés les contraintes de dépassement relatif de maximum 10% et d'un temps de réponse de 2s. Le calcul de K_i et K_p est effectué avec $K = 1$ et $\tau = 0.5$. On obtient avec les équations précédentes $K_p = 1$ et $K_i = 5.723$.

A.2 Synthèse $\mathcal{H}_\infty$ structuré

On cherche à présent à effectuer la synthèse $\mathcal{H}_\infty$ du système à l'aide de la fonction `systune` pour obtenir les mêmes performances et vérifier que l'on obtient bien des gains proches de ceux trouvés précédemment.

A.2.1 Définition sur Matlab

Pour définir les gains à calculer nous utilisons la fonction `realp`. Il faut définir chaque gain comme une fonction de transfert d'un coefficient de type `realp` puis définir les noms de l'entrée et de la sortie du bloc.

Utilisation de la fonction `realp` :

Cette fonction permet de définir des paramètres variables dont la valeur de base doit être définie. La commande `a=realp('a',2)` définit ainsi le paramètre variable a de valeur de départ 2. Un autre point fort de cette fonction pour le co-design est de pouvoir définir les bornes minimale et maximale de variation pour le paramètre avec par exemple les commandes `a.Minimum=0` et `a.Maximum=5`. Dans notre cas on peut alors définir les gains du contrôleur et le bloc G :

```
s = tf('s');

Kp   = tf(realp('Kp',0));
Kp.u = {'W'};
Kp.y = {'K_p'};

Ki   = realp('Ki',0)/ s; %Pas de fonction tf ici car Ki est defini
    comme un integrateur
Ki.u = {'err'};
Ki.y = {'K_i'};

G   = K/(tau*s+1);
G.u = {'U'};
G.y = {'W'};
```

Il est important que chaque entrée (u) et chaque sortie (y) soient définies et correspondent bien à un lien du schéma bloc. Cela définit les blocs de notre système. Ensuite nous devons définir les blocs «somme» :

```
S1 = sumblk('err = ref - W');
S2 = sumblk('U = K_i - K_p');
```

Il faut que les noms apparaissent dans les entrées/sorties des blocs définis auparavant pour que la fonction `connect` puisse fonctionner. Dès lors on utilise la commande `connect` pour créer le système dans sa globalité :

```
T=connect(Kp,Ki,G,S1,S2,{'ref'},{'W'});
```

Il faut entrer en argument les différentes fonctions de transfert crées (ou éventuellement un système de type *ss* provenant d'une linéarisation par exemple), puis les sommes, les entrées du système et enfin ses sorties. Il est possible d'ajouter d'autres paramètres mais nous verrons cela plus loin. Le système est à présent défini et il ne reste plus qu'à trouver les valeurs des gains pour respecter les contraintes.

A.2.2 Utilisation de `systune`

On commence par définir les Requirements de la fonction `systune` c'est à dire les contraintes que l'on veut imposer. Dans notre cas il s'agit de faire correspondre la réponse du système à celle d'un ordre 2 de temps de réponses à 5% de 2s et un dépassement inférieur à 10%. Le Requirement correspondant est le *StepTracking* qui permet de rapprocher la réponse à celle d'un ordre 1 ou 2.Pour plus de détails sur les Requirements de `systune`, ceux-ci sont disponibles en tapant *help TuningGoal* dans la console de Matlab.

```
TrackReq = TuningGoal.StepTracking({'ref'},{'W'},0.3,10);
TrackReq.RelGap = 0.05;
```

On met en argument les noms de l'entrée et de la sortie dont veut fixer la réponse, puis la période propre souhaitée (arrondie ici) et enfin la valeur du dépassement relatif (en pourcentage). Le facteur *RelGap* donne l'écart à viser entre les réponses réelle et souhaitée, il vaut ici 5%.

On utilise alors `systune` :

```
ST = systune(T,TrackReq);
```

On a en entrée le système connecté et le Requirement imposé.

On peut extraire la fonction de transfert entre la Référence et la sortie par la fonction :

```
T2 = getIOTransfer(ST,{'ref'},{'W'});
```

De même la valeur des blocs obtenus par l'optimisation peuvent être extrait par :

```
B = getBlockValue(ST)
```

On trace alors la comparaison des réponses entre les valeurs trouvées précédemment et celles fournies par `systune` :

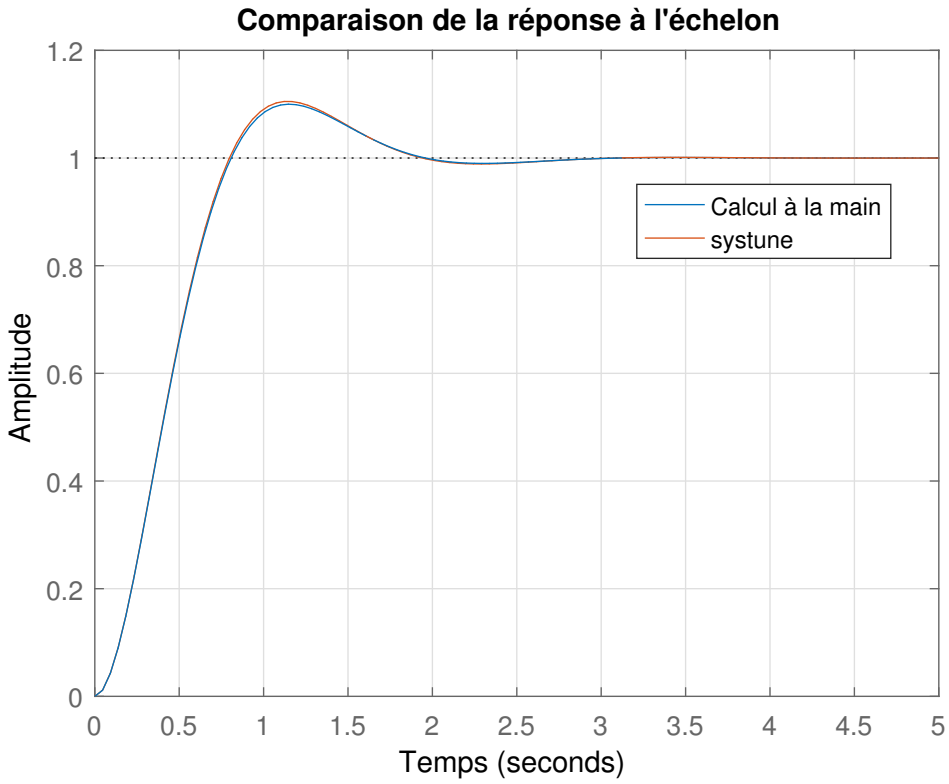

Figure A.2 Comparaison entre la simulation et le résultat précédent

On voit très clairement que les deux courbes sont très semblables (l'écart est dû à une approximation de la période propre dans la formule du `StepTracking`). De même, on exporte les valeurs des blocs K_p et K_i qui donnent des valeurs très proches de celles obtenues à la main :

$$K_p = 0.9705$$
$$K_i = 5.5556$$

A.3 Co-design

Nous cherchons à présent à obtenir les mêmes performances tout en augmentant la constante de temps du moteur τ. Pour cela on la définit également comme un paramètre variable du problème et on contraint ce coefficient pour le forcer à augmenter.

Pour cela on change la définition de G dans le code Matlab :

```
tau          = realp('tau',0.5);
tau.Minimum = 0.5;

G    = K/(tau*s+1);
G.u = {'U'};
G.y = {'W'};
```

De plus pour forcer le paramètre τ à augmenter, on crée un bloc fictif :

```
P    = tf(1/tau);
P.u = 'Pu';
P.y = 'Py';
```

On rajoute alors un nouveau requis :

```
Req2 = TuningGoal.Gain('Pu','Py',1);
```

Ici on commande de garder le rapport $\frac{1}{\tau}$ inférieur à 1, ce qui a pour effet d'augmenter τ lors des itérations de l'algorithme.

Remarque sur la fonction `Gain` :

Cette fonction permet de définir la valeur maximale de la fonction de transfert entre l'entrée (1^{er} argument) et la sortie (2^{e} argument). La valeur définie comme maximum doit être définie en rapport d'amplitude et non en décibel. Pour utiliser un gain en dB il faut entrer la formule sous la forme $10^{dB/20}$. Il faut ensuite entrer ceci dans la commande `systune` :

```
ST2 = systune(T,Req2,TrackReq,opt);
```

L'ordre des requis dans cette commande n'est pas anodin, il faut les trier selon 2 catégories : les `SoftGoals` et les `HardGoals`.

Utilisation de `systune` :

Comme nous l'avons déjà vu cette fonction permet d'obtenir les gains définis dans le système T de telle sorte qu'il respecte les requis imposés. La fonctions se présente sous la forme

`systune(T, SoftGoals, HardGoals, opt)`. La principale différence entre un `HardGoals` et un `SoftGoals` est que la fonction `systune` va tenter de respecter au mieux la contrainte des `SoftGoals` mais en assurant en priorité que les `HardGoals` sont bien respectés. La fonction `opt` contient ensuite différentes options associées à la commande `systune` que l'on peut trouver en utilisant `help`.

Dans notre cas, on met bien la contrainte sur le temps de réponse en `HardGoals` tandis que la contrainte sur τ n'est qu'un `SoftGoals`.

On peut à présent lancer la simulation ou bien spécifier des contraintes qui vont restreindre les valeurs possibles des gains. Il peut y avoir plusieurs types d'approche selon les limites que l'on se fixe, par exemple :

A.3.1 Contrainte sur les valeurs des gains

Si par exemple on choisit que les valeurs des gains ne peuvent pas dépasser le double des valeurs précédemment trouvées on peut changer les bornes des gains comme suit :

```
Kpc           = realp('Kpc',0)
Kpc.Maximum   = 2*Kp;
Kpc           = tf(Kpc);
Kpc.u         = {'W'};
Kpc.y         = {'K_p'};

Kic           = realp('Kic',0);
Kic.Maximum   = 2*Ki;
Kic           = Kic/s;
Kic.u         = {'err'};
Kic.y         = {'K_i'};
```

On se fixe comme limite de ne pas avoir le coefficients K_q et K_i plus de deux fois supérieurs aux valeurs trouvées précédemment. On peut vérifier que les performances temporelles sont bien respectées encore une fois :

Les valeurs des blocs K_q et K_i et du coefficient τ sont alors (Elles varient légèrement d'une simulation à l'autre) :

$$K_p{=}2$$
$$K_i{=}8.5349$$
$$\tau{=}0.7391$$

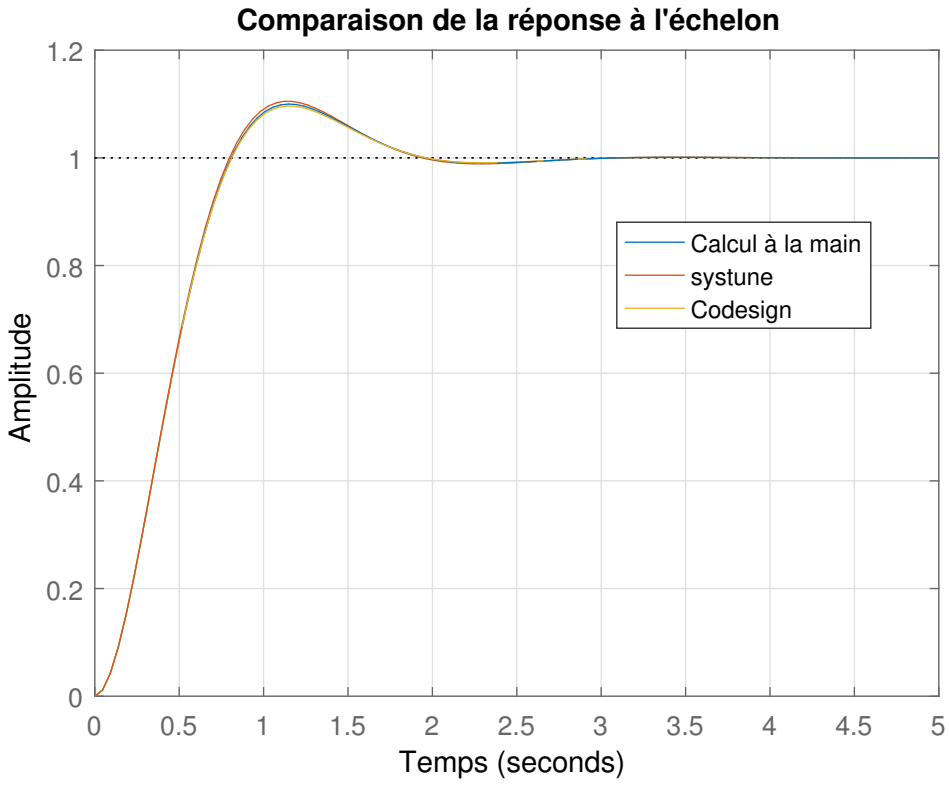

Figure A.3 Comparaison entre les différentes simulations

On a bien les coefficients inférieurs au double des valeurs précédentes et la même réponse temporelle, cependant la commande en tension du moteur augmente en contrepartie :

A.3.2 Contrainte sur l'amplitude de la commande

En pratique, on contraint plus fréquemment sur la valeur de la commande en entrée des actionneurs. Dans notre cas nous allons chercher le moteur le plus lent possible mais tout de même restreindre la tension en entrée U à 2V. Dans ce cas, on ne définit pas de bornes maximum pour les gains. Afin de contrôler l'amplitude de commande il faut définir un `Analysis Point` à ce niveau.

Les commandes correspondantes (il faut modifier la fonction `connect`) sont :

```
U=AnalysisPoint('U');
T=connect(Kpc,Kic,G,S1,S2,P,{'ref','Pu'},{'W','Py'},{'U'});
```

Dès lors on peut ajouter un nouvel Requirement sur la tension U :

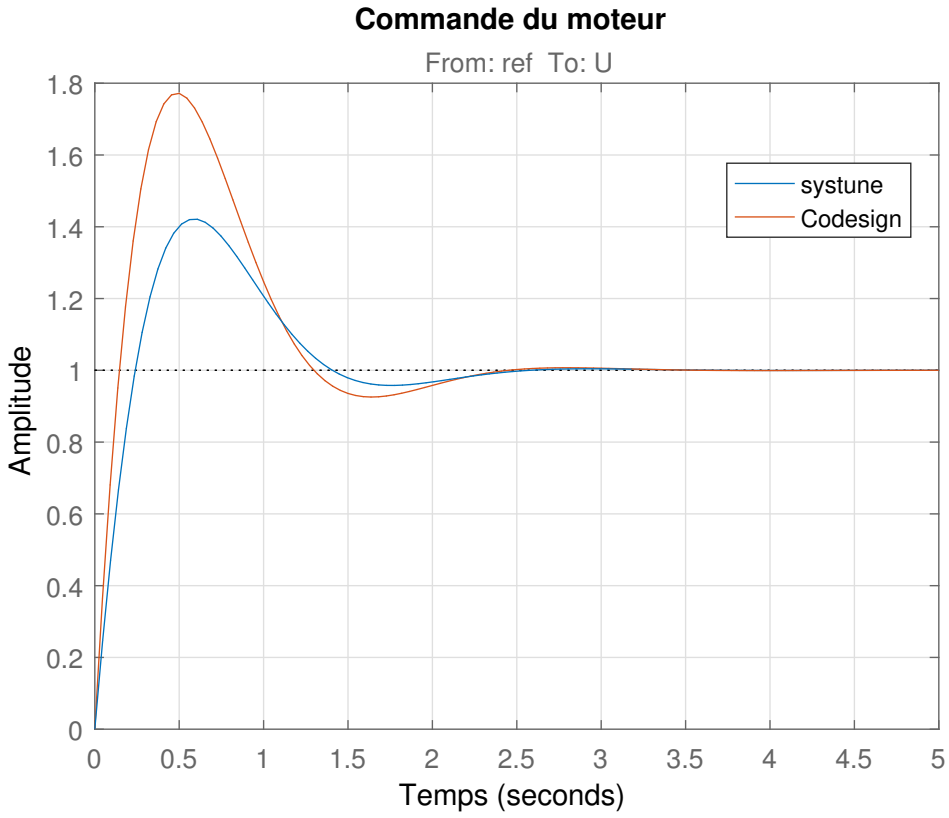

Figure A.4 Commande des moteurs

```
Req3 = TuningGoal.Gain('ref','U',10^(8.3/20));
```

La valeur du gain maximum a été trouvée à la main après plusieurs essais.

Celui-ci va être défini comme `HardGoals` afin d'assurer que le commande ne soit pas trop forte.

```
HardGoals = [Req3,TrackReq];
SoftGoals = [Req2];

opt=systuneOptions('RandomStart',10);

ST2 = systune(T,SoftGoals,HardGoals,opt);
```

On peut à présent comparer les réponses temporelles :

Un léger écart s'est creusé entre les deux réponses mais le temps de réponse reste identique seule la valeur du dépassement est atténuée ce qui n'est pas un problème ici. Les valeurs des

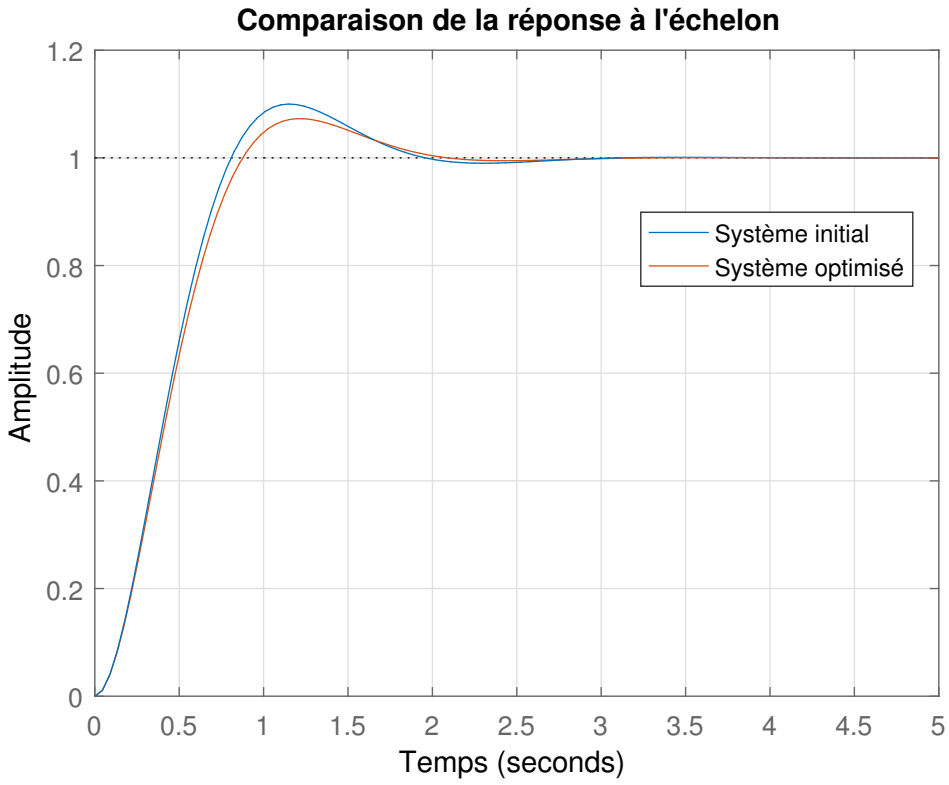

Figure A.5 Comparaison entre simulations

blocs K_q et K_i et du coefficient τ sont à présent :

$$K_p{=}3.0570$$
$$K_i{=}10.6658$$
$$\tau{=}0.9413$$

La constante de temps est à présent quasiment le double de sa valeur initiale pour une réponse comparable ce qui prouve bien l'efficacité de la démarche appliquée.
On vérifie à présent que la tension U ne dépasse pas 2V comme attendu :

En conclusion, nous avons bien été capables de trouver un moteur presque deux fois plus lent avec lequel on obtient la même réponse.

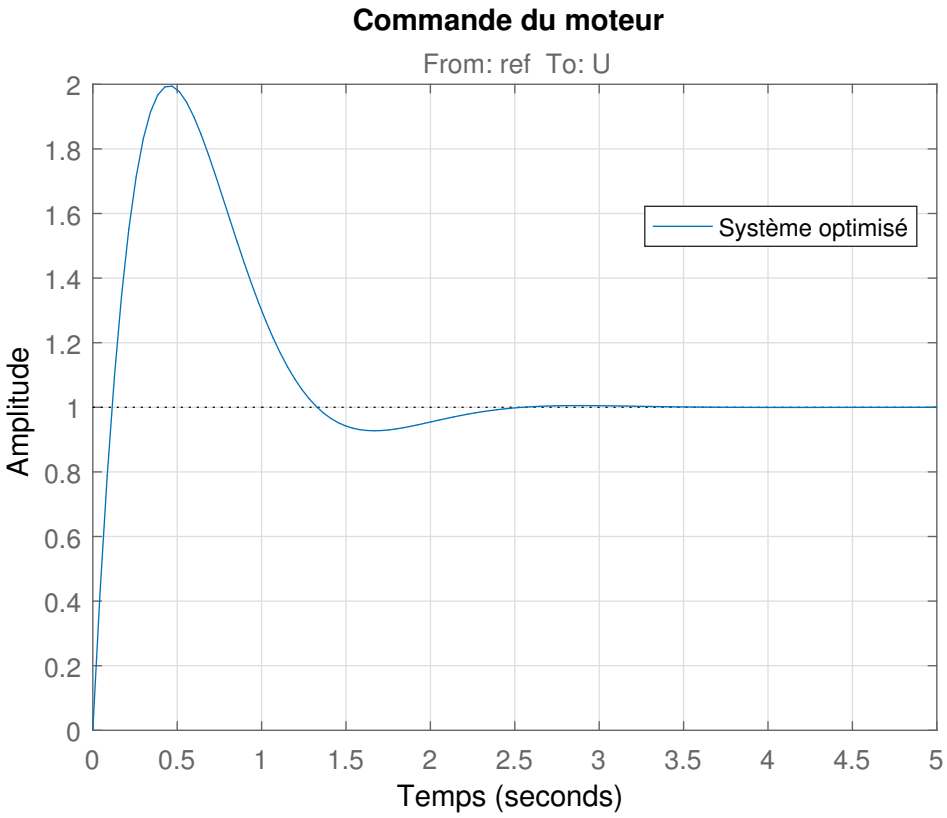

Figure A.6 Tension U du moteur